PHYSWAM: PHYSICALLY CONSISTENT WORLD–ACTION MODELING FOR AUTONOMOUS DRIVING

Anonymous authors
Paper under double-blind review

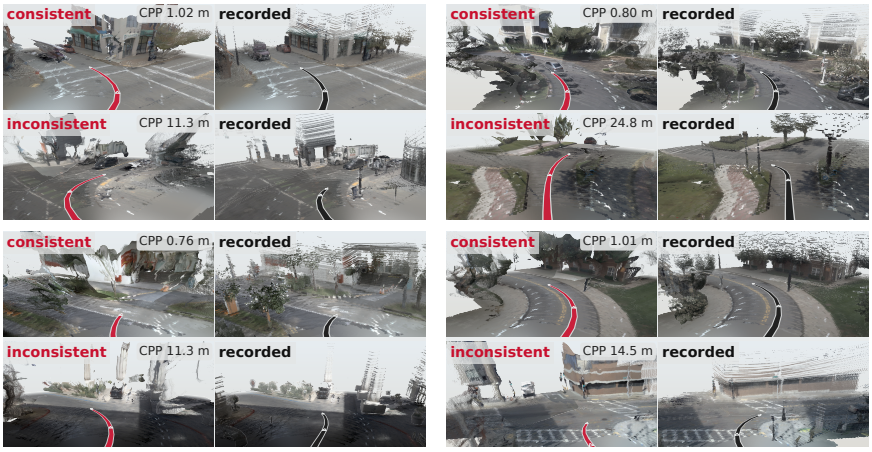

Figure 1: **Physical consistency through Coupled Point Projection (CPP).** CPP jointly supervises generated depth and ego motion against measured scene geometry. Generated depth is unprojected into 3D and transformed using generated motion (left). LiDAR observations transformed using recorded motion provide the reference in the same coordinate frame (right). The resulting geometric loss supervises both predictions: errors in depth, motion, or both can distort or displace the generated scene relative to the measured reference. Red and black paths show generated and recorded motion, respectively. First and third rows show close alignment with the measured scene; second and fourth rows show larger geometric discrepancies.

ABSTRACT

World–action models (WAMs) jointly predict how a scene will evolve and how an agent should act, however joint generation alone does not necessarily impose a shared geometric constraint on these predictions. We present PHYSWAM, a unified world–action model for autonomous driving that co-denoises multiview video, metric depth, and ego motion within a single flow-matching transformer. To ground world and action generation in measured scene geometry, we introduce Coupled Point Projection (CPP) that unprojects the generated depth into 3D points, transforms them using the generated $\mathrm{SE}(3)$ ego motion, and minimizes their distance to LiDAR points transformed using the recorded ego motion. This geometric constraint promotes physical consistency with the measured scene by jointly supervising generated depth and motion alongside their standard flow-matching objectives. At inference, trajectory selection relies only on a simple label-free consensus rule, with no learned scorer or simulator feedback. We evaluate PHYSWAM across NAVSIM v1 and v2 planning, zero-shot closed-loop transfer, and future video and metric-depth prediction. Despite PHYSWAM's simple selection procedure, it achieves strong planning performance and transfers zero-shot to unseen driving environments. It also generates accurate metric depth and temporally coherent video, with CPP improving both planning and depth prediction. Together, these results demonstrate that the geometric relationship between scene depth and ego motion provides a direct way to couple world and action generation within a simple unified model.

1 Introduction

A driving decision changes both how the ego vehicle moves and what it observes along its route. A planned trajectory and the scene observed along it therefore describe the same future from different perspectives. End-to-end driving systems learn such decisions directly from sensor observa-

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.

tions [42; 21; 24; 34], driving world models generate future scenes [55; 19; 56; 16], increasingly across views and with geometric conditioning [15; 26], and recent world–action models predict future observations and ego motion together [66; 38; 45]. Joint generation alone, however, does not impose a shared geometric constraint on these predictions.

Existing approaches connect scene prediction and planning by using predicted images or features to shape a policy's representation [28; 30; 52; 70], by exposing present or predicted geometry to the planner [81; 67; 40], by adding depth and other scene targets alongside video and action learning [76; 44; 59], or by supervising the generated video with geometry [14]. These approaches establish the value of scene prediction for driving. The geometric relationship between scene depth and ego motion also lets us supervise the scene they describe together, using measured geometry as a common reference. This motivates a shared objective that depends on both generated depth and motion, complementing their individual prediction losses. Table 17 in App. E summarizes how representative works connect future prediction to planning. This naturally motivates the question: *how can we use the geometric relationship between scene depth and ego motion to couple world and action generation?*

Metric depth offers a direct way to express this relationship: feed-forward reconstruction methods recover scene structure and camera geometry from images [54; 49; 25], and dynamic reconstruction methods recover geometry and motion over time from video [65; 64]. We draw on this geometric perspective to connect world and action generation. Generated depth can be unprojected into 3D and transformed using generated ego motion into a shared reference frame. LiDAR observations transformed using recorded motion provide the measured scene geometry in that same frame. Errors in depth, motion, or both can distort or displace the generated scene relative to this reference. We promote physical consistency with the measured scene by penalizing this discrepancy through a single geometric loss that supervises both predictions. Figure 1 visualizes this comparison for generated futures with small and large geometric discrepancies.

We introduce PHYSWAM, a unified world–action model that co-denoises multiview video, metric depth, and ego motion within a single transformer. Building on the Cosmos family of pretrained world models [1; 2], we represent the three modalities in a shared sequence trained with flow matching [36]. Conditioned on current multiview images, camera calibration, driving context, and ego history, the model jointly generates future video, metric depth maps, and an ego trajectory through iterative denoising. Bidirectional attention allows all three modalities to exchange information throughout this process.

To explicitly couple the depth and ego motion generated by this model, we introduce *Coupled Point Projection* (CPP). CPP unprojects generated depth into 3D and transforms the resulting points using generated $\mathrm{SE}(3)$ ego motion. Corresponding LiDAR observations transformed using recorded motion provide the reference in the same coordinate frame. CPP penalizes distances between the generated and measured geometry, so errors in depth, motion, or both contribute to the same loss. The supervision of each prediction therefore depends on the other, complementing their individual flow-matching objectives. This promotes physical consistency with the measured scene through the geometric relationship between generated depth and ego motion. We also keep inference simple. Several driving systems use learned scorers to evaluate candidate trajectories, either directly or from predicted future states [31; 46; 29; 73]. For each input scene, PHYSWAM instead samples multiple trajectories and selects a representative one using only their pairwise distances. This parameter-free consensus rule requires no labels, learned scorer, or simulator feedback.

We evaluate PHYSWAM both as a driving policy and through the quality of its generated future world. Planning experiments cover NAVSIM v1 and v2, including `navtest` and two-stage pseudo-simulation [12; 6]. Motivated by recent evidence of cross-domain driving transfer [38; 51], we also evaluate zero-shot closed-loop driving in HUGSIM [74], without target-domain fine-tuning. Beyond planning, we assess future-video quality and metric-depth accuracy. Despite its simple trajectory selection, PHYSWAM achieves strong planning compared to leading methods and transfers zero-shot across driving environments. It also generates accurate metric depth and temporally coherent video. CPP improves both planning and depth prediction, while consensus selection improves planning over using a single generated trajectory for each input scene. Together, these results demonstrate the value of coupling generated depth and ego motion through a shared geometric objective.

Our contributions are threefold. **Unified world–action generation:** we develop a shared flow-matching formulation that co-denoises multiview video, metric depth, and ego motion within one model. **Physical consistency through geometric coupling:** we introduce Coupled Point Projection (CPP), a geometric objective that jointly supervises generated depth and ego motion against measured scene geometry, complementing their individual prediction objectives. **Broad evaluation**

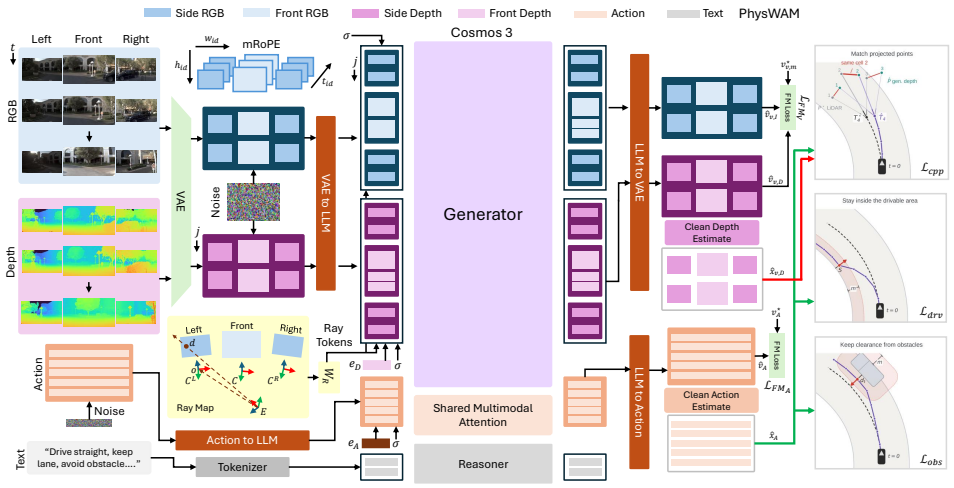

Figure 2: **Overview of PhysWAM.** The Cosmos 3 generator [2] jointly denoises multiview video, metric depth and ego motion, conditioned on text through its frozen reasoner; RGB and depth share a frozen VAE, and calibrated ray embeddings with mRoPE encode camera geometry and position. Current RGB and motion history stay clean; the rest is noised. Flow matching supervises the velocity outputs, whose clean estimates feed CPP, which transforms generated depth with generated motion and compares it with LiDAR under recorded motion, and the two hinges.

with simple inference: we demonstrate strong planning and zero-shot closed-loop transfer with label-free trajectory selection, alongside accurate metric depth and temporally coherent video.

2 Related Work

Generative driving world models. Driving world models generate controllable future video [55; 19; 16], across cameras in GAIA-2 [43], and with LiDAR, occupancy, depth or semantics added to the generated future in MUVO, OccWorld and OmniNWM [4; 71; 26]. PhysWAM generates multiview video and metric depth together with ego motion in one flow-matching model.

World–action models and planning. Future prediction improves policies as an auxiliary objective [28; 70] and through joint video–action generation [66; 38; 79; 63]; predicted futures also serve learned trajectory evaluation, refinement and safety-aware exploration [72; 60; 37; 44]. PhysWAM follows joint generation but selects among samples with a parameter-free consensus rule, without a learned scorer or simulator feedback.

Geometry in world and action generation. Geometry enters generative driving models as a conditioning signal along supplied trajectories [7; 62], as a generated output [32; 17; 76], as a planning representation [67; 40], or as a training target for the video features [57; 14; 49; 50], a line that goes back to learning depth and camera motion from view synthesis [75]. PhysWAM adds generated ego motion to the geometric objective: CPP unprojects the generated depth, carries it with the generated motion, and compares it with LiDAR under the recorded motion in one frame, supervising both outputs alongside their flow-matching objectives (App. E discusses these works in more detail).

3 Method

PhysWAM combines joint video, depth, and motion generation with a geometric objective that supervises the depth–motion pair against measured scene structure (Fig. 2). We first define the generation problem and its representation, then introduce CPP, supporting trajectory constraints, and the training and inference procedures.

3.1 Problem Formulation and Preliminaries

World–action modeling with metric geometry. We study a driving world–action model that generates future observations and ego motion together with explicit scene geometry. Let $\mathcal{V} = \{\mathrm{F}, \mathrm{L}, \mathrm{R}\}$ denote the front, left, and right cameras, and let $t = 0, \ldots, T$ index scene frames, with $t = 0$ the current time. For view v, $\mathbf{I}_t^v$ is an RGB image and $\mathbf{D}_t^v$ is optical-axis depth in metres. The context c contains current images $\{\mathbf{I}_0^v\}_{v\in\mathcal{V}}$, camera calibration, textual driving context, and recent ego motion $\mathbf{a}_0$. Starting from a pretrained world model [2], we learn a conditional generative model $p_\theta(\{\mathbf{I}_{1:T}^v, \mathbf{D}_{0:T}^v\}_{v\in\mathcal{V}}, \mathbf{a}_{1:T} \mid c)$, where $\mathbf{a}_t$ represents relative ego motion ending at time t. Depth is generated at both current and future times; current RGB, motion history provide clean conditioning.

Coordinate conventions. The front camera is the reference view: $\mathcal{C}_t$ is its frame, $\mathcal{C}_t^v$ the frame of view v, $\mathcal{E}_t$ the ego frame, and $\mathbf{T}_{A\leftarrow B}$ maps coordinates from B to A. The camera-to-ego calibration $\mathbf{G}_v = \mathbf{T}_{\mathcal{E}_t\leftarrow\mathcal{C}_t^v}$, fixed within a sequence, gives $\mathbf{E}_v = \mathbf{G}_{\mathrm{F}}^{-1}\mathbf{G}_v$, which maps view v into the front camera at the same time. Each relative motion is $\mathbf{\Delta}_t = \mathbf{T}_{\mathcal{C}_{t-1}\leftarrow\mathcal{C}_t}$, and the composition $\mathbf{T}_t = \mathbf{\Delta}_1 \cdots \mathbf{\Delta}_t$, with $\mathbf{T}_0 = \mathbf{I}$, places the future front-camera frame in $\mathcal{C}_0$.

Conditional flow matching. We parameterize p_θ with the generative transformer of a pretrained world model [2] and train it using rectified flow matching [39]. Let $\mathbf{x}$ collect the continuous representations to be generated: VAE latents for RGB and depth, and pose coordinates for motion, defined in Sec. 3.2. At noise level $\sigma \in [0, 1]$,

$$\mathbf{x}^\sigma = (1-\sigma)\mathbf{x} + \sigma\boldsymbol{\epsilon}, \qquad \mathbf{v}^\star = \boldsymbol{\epsilon} - \mathbf{x}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{1}$$

One noise level is shared across the generated modalities of an example, with independently drawn Gaussian entries. The transformer predicts the joint velocity $\widehat{\mathbf{v}} = v_\theta(\mathbf{x}^\sigma, \sigma; c)$. These operations apply only to generated entries: conditioning frames and motion history are kept clean. The estimated clean representation is

$$\widehat{\mathbf{x}} = \mathbf{x}^\sigma - \sigma\widehat{\mathbf{v}} = \mathbf{x} + \sigma(\mathbf{v}^\star - \widehat{\mathbf{v}}). \tag{2}$$

This identity connects velocity prediction to the geometric objectives in Secs. 3.3–3.4: they act on clean depth and motion estimates from current forward pass, without complete denoising rollout.

3.2 Unified World–Action–Geometry Denoising

Model architecture. We adapt the Cosmos 3 Mixture-of-Transformers architecture [2; 33], based on Qwen3-VL [3]. Its understanding and generation pathways have separate parameters with shared attention: text attends causally within the text context, while visual and action tokens attend to text and one another bidirectionally. We fine-tune the generation pathway, its projections, and embeddings, keeping the text pathway and video VAE fixed. We extend this architecture to jointly denoise multiview video, depth, and motion without a separate depth denoiser.

Visual and motion representations. The frozen causal video VAE [48], with encoder $\mathcal{E}$ and decoder $\mathcal{D}$, yields $\mathbf{x}_{v,I} = \mathcal{E}(\mathbf{I}_{0:T}^v)$ and $\mathbf{x}_{v,D} = \mathcal{E}(\mathcal{C}(\mathbf{D}_{0:T}^v))$ (depth labels: App. B.2). The fixed encoding $\mathcal{C}$ maps clipped log depth to the image range and repeats it across channels, preserving a common conversion to metres. For both modalities, side-view height and width are half the front view's, reducing tokens while retaining lateral coverage. Visual latents have shape $J \times h_v \times w_v \times c_\ell$: $j = 0, \ldots, J-1$ indexes latent frames, h_v, w_v are spatial dimensions, and c_ℓ is the channel width. Temporal compression s_t associates latent frame j with scene frame $\kappa(j) = s_t j$, including $j = 0$ for the current frame. Actions retain the pretrained pose representation [2], $\mathbf{x}_A = \mathbf{a}_{0:T}$ with $\mathbf{a}_t = [\mathbf{t}_t; \mathbf{r}_t^{(1)}; \mathbf{r}_t^{(2)}]$ and each component in $\mathbb{R}^3$. Translation is unnormalized in metres in the preceding front-camera frame; Gram–Schmidt orthonormalization of the first two rotation columns [77] gives the relative transform $\mathbf{\Delta}(\mathbf{a}_t)$.

Tokenization and camera conditioning. A visual stream $i = (v, m)$ identifies view v and modality $m \in \{I, D\}$; A denotes the action stream. The operator $\mathcal{P}$ patchifies (flattens) spatial $p \times p$ patches of VAE cells $u = (u_h, u_w)$ into $p^2 c_\ell$-dimensional vectors, with each patch $q = (q_h, q_w)$ mapped to one token. Noise and velocity targets remain in $\mathbf{x}_i$; projection yields $\mathbf{z}_i \in \mathbb{R}^{N_i \times d_h}$, containing N_i tokens of hidden width d_h. For patch q, its image-region centre $\boldsymbol{\xi}_v(q)$ is determined by the VAE stride and patch size. Calibration gives its unit viewing direction $\mathbf{d}_v(q)$ and camera centre $\mathbf{o}_v$ in the instantaneous ego frame. The Plücker ray $\mathbf{r}_v(q) = [\mathbf{d}_v(q); \mathbf{o}_v \times \mathbf{d}_v(q)]$ conditions projected patch:

$$\mathbf{z}_{j,q}^i = \mathbf{W}_V[\mathcal{P}(\mathbf{x}_i^\sigma)]_{j,q} + \mathbf{b}_V + \mathbf{W}_R\mathbf{r}_v(q), \qquad i = (v, m). \tag{3}$$

The ray embedding is shared across RGB and depth and repeated over time; depth additionally receives a learned embedding $\mathbf{e}_D$. Actions use $\mathbf{z}_t^A = \mathbf{W}_A \,\mathrm{pad}_A(\mathbf{a}_t^\sigma) + \mathbf{b}_A + \mathbf{e}_A$, reusing the pretrained pose projection and embedding [2]. The operator pad_A zero-pads to the projection's input width; padding coordinates are excluded from losses. All generated tokens receive the shared noise embedding $\mathbf{e}_\sigma(\sigma)$; clean context tokens do not. The new $\mathbf{W}_R$ and $\mathbf{e}_D$ are zero-initialized to preserve pretrained content embeddings.

Spatial and temporal alignment. Following the pretrained model [2], mRoPE [53] encodes time, height, and width in attention queries and keys. For scene frame rate f, let $\tau(k) = \tau_0 + \alpha k/f$, with common offset τ_0 after text and pretrained time scale α in coordinate units per second. We assign

$$\boldsymbol{\rho}_{j,q}^{v,I} = \boldsymbol{\rho}_{j,q}^{v,D} = \big(\tau(\kappa(j)),\, q_h + b_v^h,\, q_w + b_v^w\big), \qquad \boldsymbol{\rho}_t^A = \big(\tau(t),\, 0,\, 0\big). \tag{4}$$

RGB and depth share coordinates; actions align with their endpoint time, with κ accounting for visual temporal compression. The offsets (b_v^h, b_v^w) arrange left–front–right token grids panoramically; calibrated rays supply camera geometry.

Velocity outputs and flow-matching objectives. From final hidden tokens $\mathbf{z}_i^{\text{out}}$, output projections return velocities in the original representation spaces, $\widehat{\mathbf{v}}_i = \mathcal{P}^{-1}(\mathcal{O}_V(\mathbf{z}_i^{\text{out}}))$ and $\widehat{\mathbf{v}}_A = \text{unpad}_A(\mathcal{O}_A(\mathbf{z}_A^{\text{out}}))$. The visual projection returns $p^2 c_\ell$ values per patch; unpatchification restores the VAE grid for computing the flow-matching loss as well as clean estimates (via Eq. (2)). Let MSE average squared velocity error within each stream, assigning zero error to conditioning entries. With view–modality weights $\omega_{v,m}$,

$$\mathcal{L}_{\text{FM},V} = \mathbb{E}\left[\sum_{v,m} \omega_{v,m}\,\text{MSE}(\widehat{\mathbf{v}}_{v,m}, \mathbf{v}^\star_{v,m})\right], \qquad \mathcal{L}_{\text{FM},A} = \mathbb{E}[\text{MSE}(\widehat{\mathbf{v}}_A, \mathbf{v}^\star_A)]\,. \tag{5}$$

Expectations average over examples, noise levels, and Gaussian noise. These losses supervise each output individually; CPP explicitly supervises their geometric relationship.

3.3 Coupled Point Projection

Depth and ego motion jointly determine how the generated scene is placed in 3D. CPP uses this relationship to supervise both predictions against measured scene geometry: it transforms generated depth with generated motion, then compares the result with LiDAR transformed using recorded motion. This gives a shared metric objective without sampling a full future during training.

Efficient metric-depth estimation. Applying geometric supervision through the full video VAE decoder would require decoding dense depth at every training update. Instead, a lightweight convolutional depth head Π_ϕ maps the estimated clean depth latents $\widehat{\mathbf{x}}_{v,D}$ directly to metric depth on the VAE grid. It applies three spatial convolutions with GELU activations to each latent frame and produces one log-depth value per cell without changing spatial resolution. Writing $\ell_{\min} = \log(1 + d_{\min})$ and $\ell_{\max} = \log(1 + d_{\max})$ for the fixed depth-encoding bounds, $\widehat{d}_j^v(u) = \exp(\text{clip}([\Pi_\phi(\widehat{\mathbf{x}}_{v,D})]_j(u), \ell_{\min}, \ell_{\max})) - 1$, where u indexes a VAE latent grid cell, not a transformer patch, and $\widehat{d}_j^v(u)$ is optical-axis depth in metres. We fit the depth head Π_ϕ offline using masked smooth L_1 regression in log space to geometric-mean LiDAR depth pooled within each valid latent cell, using both clean depth latents and estimated-clean latents sampled from the training noise distribution. Its parameters are then kept fixed during world model training, while CPP gradients pass through Π_ϕ to supervise the generated depth latents. Clean action estimates $\widehat{\mathbf{a}}_t$ are converted to relative transforms and composed as $\widehat{\mathbf{T}}_t = \boldsymbol{\Delta}(\widehat{\mathbf{a}}_1) \cdots \boldsymbol{\Delta}(\widehat{\mathbf{a}}_t)$.

Generated and measured geometry. Let $\mathbf{b}_v(u)$ be the unit camera-frame ray through the image location corresponding to latent cell u. Calibrated unprojection is $\mathcal{U}_v(d, u) = d\,\mathbf{b}_v(u)/b_{v,z}(u)$: dividing by the ray's optical-axis component makes its z coordinate one, so multiplication by optical-axis depth d gives the camera-frame 3D location. These rays are evaluated at VAE-cell centres; the ray embeddings in Sec. 3.2 use the centres of the larger token patches. We evaluate CPP at future anchor frames $\mathcal{A} \subseteq \{\kappa(j) : j > 0\}$ and denote their latent indices by $j(t) = \kappa^{-1}(t)$. The $t = 0$ (or $j = 0$) frame is treated as the reference for this supervision. For each anchor, let $d_t^{L,v}(u)$ be the geometric mean of the LiDAR depths projected into cell u, and let $\mathbf{T}_t^\star$ be the recorded front-camera pose in $\mathcal{C}_0$. Generated and measured depths use the same calibrated cell ray:

$$\widehat{\mathbf{P}}_{v,t}(u) = \widehat{\mathbf{T}}_t \mathbf{E}_v\, \mathcal{U}_v\Big(\widehat{d}^v_{j(t)}(u), u\Big), \qquad \mathbf{P}^\star_{v,t}(u) = \mathbf{T}^\star_t \mathbf{E}_v\, \mathcal{U}_v\Big(d_t^{L,v}(u), u\Big). \tag{6}$$

At each anchor, collecting these locations across LiDAR-supported cells and views forms two point clouds in $\mathcal{C}_0$: the scene implied by generated depth and motion, and its measured counterpart. Corresponding cells identify the locations compared by CPP. Both clouds describe the same future time, so expressing them in $\mathcal{C}_0$ does not require moving objects to remain static across the sequence. Calibration, LiDAR depths, and recorded poses are fixed supervision (App. B.3).

Physical consistency through geometric coupling. Let Ω contain valid (v, t, u) correspondences with LiDAR support. We use a robust metric loss

$$\mathcal{L}_{\text{cpp}} = \frac{1}{|\Omega|} \sum_{(v,t,u)\in\Omega} \rho_\delta\Big(\big\|\widehat{\mathbf{P}}_{v,t}(u) - \mathbf{P}^\star_{v,t}(u)\big\|_2\Big), \tag{7}$$

where ρ_δ is Huber penalty [23], normalized by its transition distance: $\rho_\delta(e) = e^2/(2\delta)$ for $e \le \delta$ and $e - \delta/2$ otherwise. This retains metric units while reducing influence of large discrepancies.

We average over valid correspondences across views and anchors, then over examples with LiDAR support. Errors in depth, motion, or both affect the same comparison. Motion supervision depends on the generated geometry being transformed, while depth supervision depends on the generated transformation. Gradients pass through the depth head and the composed action transforms to the corresponding clean estimates. CPP therefore promotes physical consistency *with the measured scene* by jointly supervising how generated depth and motion represent it. The flow-matching objectives retain direct supervision of the individual predictions.

3.4 Training and Inference

Geometric hinge losses. CPP supervises the generated depth–motion pair; two complementary hinge losses supervise waypoint positions through the generated motion alone, independently of generated depth. Both act on the generated front-camera position $\widehat{\mathbf{p}}_t = \pi_{xz}(\mathrm{trans}(\widehat{\mathbf{T}}_t))$ projected onto the ground plane of $\mathcal{C}_0$, against obstacle and drivable-area labels placed in the same frame (App. B.4). The obstacle term $\mathcal{L}_{\mathrm{obs}}$ is a squared hinge on the clearance of $\widehat{\mathbf{p}}_t$ from static LiDAR structure and annotated object footprints at time t: it penalizes clearance below a margin m_{obs}, capped at the clearance the recorded trajectory kept, so that the recorded path incurs no loss even in narrow corridors. The drivable-area term $\mathcal{L}_{\mathrm{drv}}$ is a squared hinge on the signed distance of $\widehat{\mathbf{p}}_t$ inside the drivable region, computed from a distance-field raster by bilinear interpolation, below a margin m_{drv}. Both margins apply to positions rather than vehicle footprints, and the labels supply training supervision only; see App. A.3 for details.

Combined objective. All geometric losses act on the clean estimates in Eq. (2). We downweight heavily noised examples with $w(\sigma) = (1-\sigma)^2$ and define noise-weighted losses $\overline{\mathcal{L}}_k = \mathbb{E}[w(\sigma)\mathcal{L}_k]$ for $k \in \{\mathrm{cpp}, \mathrm{obs}, \mathrm{drv}\}$. The combined objective is

$$\mathcal{L}_{\mathrm{total}} = \lambda_V \mathcal{L}_{\mathrm{FM},V} + \lambda_A \big(\mathcal{L}_{\mathrm{FM},A} + s_{\mathrm{cpp}}\overline{\mathcal{L}}_{\mathrm{cpp}} + s_{\mathrm{obs}}\overline{\mathcal{L}}_{\mathrm{obs}} + s_{\mathrm{drv}}\overline{\mathcal{L}}_{\mathrm{drv}}\big), \tag{8}$$

where $\lambda_V, \lambda_A > 0$ weight visual and action supervision and the s_k balance the geometric terms on the current update.

Output-gradient balancing. We balance gradient magnitudes [9] rather than loss values. On every update, s_k is set so that the gradient of each active geometric term at the action velocity output has a fixed fraction η_k of the norm of the action flow-matching gradient, and a second gain on CPP's backward depth path imposes the same fraction of the depth flow-matching gradient at the depth output, leaving forward values and the action path unchanged. The coefficients are measured with local backward passes to the velocity outputs, held fixed during the single full backward pass, and set to zero for inactive terms (App. A.4).

Joint sampling and trajectory selection. At inference, we jointly denoise Gaussian initializations from $\sigma = 1$ to 0 with current RGB, motion history, and text fixed; the VAE decoder returns RGB and metric depth, and composed motion, mapped to ego coordinates, gives the trajectory. From K samples we select the medoid trajectory, the one with the smallest total planar distance to others (App. A.5). This parameter-free rule uses no measured future, learned scorer, or simulator feedback.

4 Experiments

Data and model. We train on the 103,281 windows of the NAVSIM `navtrain` split [12; 11]: the current frame and the eight following frames at 2 Hz from the front, front-left, and front-right cameras (App. B). Depth, LiDAR, obstacle, and drivable-area labels are used in training only; at inference the model receives images, calibration, text, and motion history. PHYSWAM is initialized from Cosmos 3 Nano [2]; its generation pathway (7.0B of 15.2B parameters) is fine-tuned and the rest of the model is fixed.

Training. We train for 30,000 updates with batches of 44 windows on NVIDIA RTX PRO 6000 Blackwell GPUs, using AdamW with a peak learning rate of 2×10^{-5} decayed linearly to zero, and evaluate an exponential moving average of the weights. In Eq. (8), $\lambda_V = 10$, $\lambda_A = 20$, and the target fraction η_k of every geometric term is 0.2. All hyperparameters are listed in App. C.1.

Benchmarks. On NAVSIM `navtest` (12,146 scenes) we report PDMS, the v1 score, and EPDMS, the v2 score, from the official devkit (sub-scores in App. C.1). On `navhard` [6] we report the two-stage pseudo-simulation score: the plan is scored in 450 recorded scenes and then in 5,462 synthetic scenes with reactive traffic (App. C.2). For zero-shot closed-loop driving we run all 436 HUGSIM episodes [74] without any target-domain training, replanning every 0.5 s, and report route completion (RC) and HD-Score, weighting the four difficulty levels by their number of episodes.

Table 1: **NAVSIM v1 and v2 on `navtest`.** Numbers of other methods are as reported in their papers. R34: ResNet-34 backbone; +L: LiDAR input. Bold: best per column, oracle row excluded. PHYSWAM rows use one sample per scene unless otherwise specified; the sample-to-sample sd is 0.24 PDMS and 0.30 EPDMS.

		NAVSIM v1						NAVSIM v2									
Method	Input	NC	DAC	TTC	C	EP	PDMS	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	EPDMS
Human	–	100	100	100	99.9	87.5	94.8						–				
End-to-end planners																	
TransFuser [10]	3×Cam+L	97.7	92.8	92.8	100.0	79.2	84.0	96.9	89.9	97.8	99.7	87.1	95.4	92.7	98.3	87.2	76.7
LTF [10]	3×Cam				–			97.6	91.9	99.2	99.8	87.6	97.2	96.6	98.3	86.3	83.6
Hydra-MDP++ (R34) [27]	3×Cam	97.6	96.0	93.1	100	80.4	86.6	97.2	97.5	99.4	99.6	83.1	96.5	94.4	98.2	70.9	81.4
ARTEMIS [13]	3×Cam+L	98.3	95.1	94.3	100	81.4	87.0	98.3	95.1	98.6	99.8	81.5	97.4	96.5	–	**98.3**	83.1
DiffusionDrive [34]	3×Cam+L	98.2	96.2	94.7	100.0	82.2	88.1	98.2	95.9	99.4	99.8	87.5	97.3	96.8	98.3	87.7	84.5
WoTE [29]	3×Cam+L	98.5	96.8	94.9	99.9	81.9	88.3						–				
BeyondDrive [51]	3×Cam	98.4	97.9	95.0	100.0	83.7	89.7	98.4	97.9	99.5	99.8	87.8	98.0	97.3	98.3	88.5	90.1
DriveSuprim (R34) [61]	3×Cam	97.8	97.3	93.6	100	86.7	89.9	97.5	96.5	99.4	99.6	88.4	96.6	95.5	98.3	77.0	83.1
Vision–language–action models																	
AutoVLA [78]	3×Cam	98.4	95.6	98.0	99.9	81.9	89.1						–				
ReCogDrive [58]	3×Cam	97.9	97.3	94.9	100	87.3	90.8						–				
World models and world–action models																	
Epona [66]	1×Cam	97.9	95.1	93.8	99.9	80.4	86.2						–				
PWM [69]	1×Cam	98.6	95.9	95.4	100.0	81.8	88.1						–				
DriveVLA-W0 [30]	1×Cam	98.7	96.2	95.5	100.0	82.2	88.4						–				
DVGT-2 [81]	8×Cam	97.8	97.2	93.9	100	83.4	88.6	97.8	97.2	99.6	**99.9**	88.4	97.3	98.1	98.2	83.2	88.9
DriveDreamer-Policy [76]	3×Cam	98.4	97.1	95.1	100.0	83.5	89.2	98.4	97.1	99.5	**99.9**	87.9	97.7	97.6	98.3	79.4	88.7
DriveVLA-W0 (anchors) [30]	1×Cam	98.7	**99.1**	95.3	99.3	83.3	90.2	98.5	**99.1**	98.0	99.7	86.4	98.1	93.2	97.9	58.9	86.1
DVGT-2-NAVSIM [81]	8×Cam	98.7	97.9	95.8	100	84.3	90.3	98.7	97.9	**99.7**	**99.9**	87.9	98.0	**98.2**	98.2	77.0	89.6
GeoWAM [40]	8×Cam				–			98.7	97.7	**99.7**	**99.9**	87.0	98.1	97.9	98.3	86.8	90.2
PHYSWAM	3×Cam	**99.0**	97.5	98.5	99.8	88.7	91.4	**99.0**	97.5	99.3	99.8	88.7	98.5	96.5	98.6	90.5	90.3
PHYSWAM, medoid-of-8	3×Cam	**99.0**	97.4	**98.6**	99.9	**89.6**	**91.7**	**99.0**	97.4	99.2	**99.9**	**89.6**	**98.6**	95.8	**98.7**	90.5	**90.4**
PHYSWAM, *oracle-of-8*	3×Cam	*99.7*	*99.3*	*99.2*	*99.9*	*91.9*	*95.3*	*99.7*	*99.3*	*99.0*	*99.8*	*91.9*	*99.2*	*95.8*	*98.7*	*90.0*	*94.1*
PHYSWAM, without CPP	3×Cam	98.2	97.0	97.6	99.8	88.0	89.6	98.2	97.0	99.0	99.8	88.0	97.6	96.1	98.6	90.5	88.4

Inference. We sample with UniPC [68] for 30 steps without guidance; each sample denoises RGB, depth, and ego motion jointly, and its eight waypoints form the plan. A plan costs 9.4 GPU-seconds per scene at 30 steps; 8 steps cost 4.4 and score within a point (App. C.6). Unless marked, a row is one sample per scene; medoid rows select among $K=8$ samples by the rule of Sec. 3.4.

4.1 Planning

We compare against three families: end-to-end planners trained on NAVSIM, including the strongest recent one, BeyondDrive [51]; vision–language–action models [78; 58]; and world–action models that plan through a generative model of the scene [66; 69; 30; 81; 76; 40; 59; 14; 18]. PHYSWAM is trained by supervised fine-tuning with the CPP and hinge terms alone; several of the strongest compared methods add reinforcement learning after imitation [78; 58; 59] or a trajectory scorer distilled from the simulator [31; 61].

NAVSIM `navtest`. With a single sample per scene, PHYSWAM reaches 91.4 PDMS and 90.3 EPDMS (Table 1), the highest reported values among methods trained without a learned trajectory scorer or reinforcement-learning post-training; the best end-to-end planners (BeyondDrive, 89.7 and 90.1), vision–language–action models (ReCogDrive, 90.8) and world–action models (DVGT-2-NAVSIM, 90.3 PDMS; GeoWAM, 90.2 EPDMS) lie below, and planners that add a scorer or RL post-training reach up to 93.5 PDMS with a ViT-L backbone [61]. The margin comes from the safety terms: no-collision (99.0) and time-to-collision (98.5) are the highest reported values under both protocols, and ego progress (88.7) is 1.4 above the best reported v1 value. Selecting the medoid of eight samples adds 0.3 PDMS; the per-scene best of the eight reaches 95.3, so the samples contain better plans than label-free selection recovers.

NAVSIM `navhard`. Under two-stage pseudo-simulation (Table 2a), PHYSWAM reaches 38.1 EPDMS with a single sample and 39.8 with the medoid of eight, the highest reported scores among world–action models (GeoWAM, 36.6; EponaV2, 36.1; 4D-WAM, 35.9) and well above the end-to-end planners of Table 2 (DiffusionDrive, 27.5; LTF, 25.1). The advantage is largest in the second stage, under reactive traffic, where its no-collision, drivable-area and time-to-collision terms are the highest reported (Table 7).

Zero-shot closed loop. On HUGSIM (Table 2b), without any training on the simulator's scenes, PHYSWAM reaches RC 48.9 and HD-Score 35.5, above UniAD, VAD, LTF, and BeyondDrive (46.2 and 34.8; as the unweighted mean over the four levels, BeyondDrive's convention, PHYSWAM reads 51.4 and 38.9). The advantage is concentrated on the easy scenarios; on the medium, hard, and extreme scenarios the best reported HD-Scores remain above PHYSWAM's (Table 2b). Drivable-area compliance is above 93.8 at every difficulty, and the score is limited by the no-collision and time-to-collision terms (Table 8).

Table 2: **`navhard` and HUGSIM.** (a) NAVSIM v2 on `navhard`, two-stage pseudo-simulation with reactive traffic; EPDMS as reported, with the sources of reprinted values and the stage scores in Table 7. (b) Zero-shot closed-loop driving on HUGSIM, as reported: route completion (RC) and HD-Score by difficulty; BeyondDrive's overall is the unweighted mean over the four levels. PHYSWAM is trained on NAVSIM only, one sample per planner call. Bold: best.

(a) NAVSIM v2 on `navhard`

Method	Input	EPDMS
LTF [10]	3×Cam	25.1
DiffusionDrive [34]	3×Cam	27.5
DriveVLA-W0 (Flow) [30]	1×Cam	24.4
DVGT-2 [81]	8×Cam	31.7
DriveFuture [18]	6×Cam	34.6
4D-WAM [14]	1×Cam	35.9
EponaV2 [59]	1×Cam	36.1
GeoWAM [40]	8×Cam	36.6
PHYSWAM	3×Cam	38.1
PHYSWAM, medoid-of-8	3×Cam	**39.8**
PHYSWAM, without CPP	3×Cam	36.2

(b) Zero-shot closed-loop driving on HUGSIM

	Easy		Medium		Hard		Extreme		Overall	
Method	RC	HD	RC	HD	RC	HD	RC	HD	RC	HD
UniAD [21]	58.6	48.7	41.2	29.5	40.4	27.3	26.0	14.3	40.6	28.9
VAD [24]	38.7	24.3	27.0	9.9	25.5	10.4	23.0	8.2	27.9	12.3
LTF [10]	68.4	52.8	40.7	24.6	36.9	19.8	25.5	8.1	41.4	24.8
BeyondDrive [51]	76.8	65.6	43.0	31.4	35.5	26.3	29.6	16.2	46.2	34.8
PHYSWAM	93.5	86.9	47.4	30.1	38.6	25.2	26.1	13.4	**48.9**	**35.5**
PHYSWAM, without CPP	92.6	85.4	44.9	26.8	37.8	23.9	25.4	12.0	47.5	33.4

Table 3: **Ablations and future depth.** (a) The full recipe with and without CPP after 4,000 updates on `navtest` (off-road, collision: share of scenes with a zero drivable-area or no-collision term; heading: median error against the recorded trajectory) and at the end of training (Tables 1–2). (b, d) LoRA setting on `navtest`, one sample per scene: objective terms added to flow matching, and the front view alone against all three views by driving command, with the full model of Table 1 for reference. (c) Generated depth of the full model against the LiDAR of the same future frame and camera, without scale alignment, four samples per scene; side views in Table 13; lower block as reported on nuScenes by GeoWAM [40].

(a) CPP at the full recipe

	PDMS	EPDMS	Off-road	Collision	Heading
4k, with CPP	80.3	78.8	9.7%	3.8%	2.7°
4k, without CPP	57.0	55.7	32.7%	9.8%	8.9°

	`navtest` PDMS / EPDMS	`navhard`	HUGSIM RC / HD
final, with CPP	91.4 / 90.3	38.1	48.9 / 35.5
final, without CPP	89.6 / 88.4	36.2	47.5 / 33.4

(b) Objective terms

Objective	PDMS	EPDMS
flow	85.7	84.3
flow + hinge	87.2	85.7
flow + CPP	87.8	86.2
flow + hinge + CPP	**88.2**	**86.5**

(c) Future metric depth against LiDAR

	+2 s		+4 s	
View	AbsRel↓	$\delta_{1.25}$ ↑	AbsRel↓	$\delta_{1.25}$ ↑
Front	0.175	0.814	0.232	0.742
Front, cells	0.144	0.856	0.194	0.795
Reported on nuScenes, front view				
Epona + DVGT	0.263	0.677	0.310	0.589
Cosmos 3 + DVGT	0.376	0.513	0.422	0.447
VGGT-World	0.329	0.553	0.357	0.497
GeoWAM	0.245	0.769	0.297	0.703

(d) Views (PDMS)

Views	All	Left	Straight	Right
single-view	87.1	81.2	90.0	82.0
multi-view	**88.2**	**83.9**	**90.3**	**84.3**
full fine-tune	91.4	87.8	92.7	90.4

4.2 Ablations

We study the design of PHYSWAM in two settings. CPP is ablated at the full recipe above by training the same model without the CPP term, reported as "without CPP" in Tables 1–2. The objective terms and the views are ablated in a lighter setting: rank-128 LoRA adapters [20] on the generation pathway, trained for 16,000 updates, with all other differences from the full recipe listed in Table 10. Each row of this setting differs from the full objective only in terms or views it names.

Coupled point projection. CPP speeds up learning to plan: after 4,000 updates, the model without CPP scores 57.0 PDMS against 80.3, leaves the drivable area in a third of the scenes rather than a tenth, and has more than three times the heading error (Table 3a, top). The gap narrows with training but does not close: the final model without CPP scores 89.6 against 91.4 PDMS on `navtest`, 36.2 against 38.1 EPDMS on `navhard`, and 33.4 against 35.5 HD-Score on HUGSIM (Table 3a, bottom), below the best prior method on each (ReCogDrive, 90.8; GeoWAM, 36.6; BeyondDrive, 34.8). It also generates worse depth and motion: its per-pixel depth error against LiDAR is 8–15% higher across views and horizons (AbsRel 0.193 against 0.175 in the front view at +2 s), and fewer of its sampled futures end in a pose that agrees with the recorded one (70.9% against 81.4%; Table 13).

Objective terms. Under the LoRA setting, the hinges add 1.5 PDMS to flow matching, CPP adds 2.1, and both together add 2.5 (Table 3b). CPP is the larger contribution, and the hinges still add 0.4 on top of it. The hinges are themselves a clean-estimate loss on composed waypoint positions, so the 1.0 that CPP adds over flow + hinge is a gain beyond trajectory-space supervision of the action. Supervising depth and motion separately against the measurements would reduce to a LiDAR depth loss, which the depth stream's flow-matching target already provides, and a composed-pose loss on the measured points, which the hinges already provide; the CPP residual is the sum of these two errors plus a term bilinear in them (App. A.2). Because the residual is shared, the gradient reaching

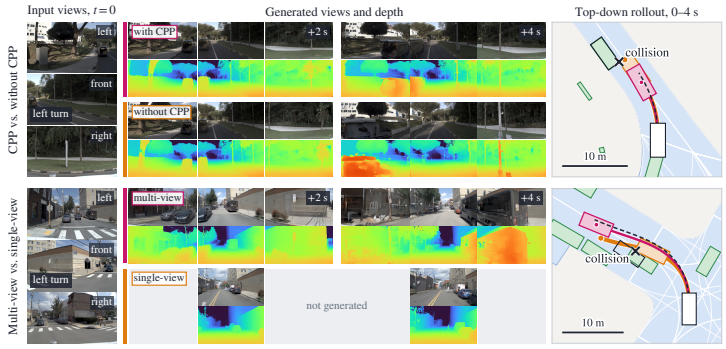

Figure 3: **Qualitative examples.** Left: input views at $t = 0$ and the command. Middle: generated views and metric depth at $+2$ and $+4$ s; the single-view model generates the front view only. Right: rollouts over the drivable area (blue), recorded traffic at 4 s (green), recorded trajectory (dashed); ego boxes at the end point or at the failure step, with the hit agent outlined. Top: with CPP the model generates a parked van a few metres off in its left view and stops short of it; without CPP it generates the van pressed against the ego and hits it at 3.4 s. Bottom: an oncoming car visible only in the left camera; the multi-view model generates it passing close and keeps the turn wide, the single-view model cuts the corner and collides at 2.3 s.

the depth carries the motion error and the gradient reaching the motion carries the depth error, so each branch is trained to agree with the other's current prediction, not only with the measurement.

Multi-view generation. Generating all three views instead of the front view alone raises PDMS by 1.1 overall (Table 3d). The gain is 0.3 on straight roads and 2.3–2.7 on turns, where the drivable boundary and crossing traffic lie in the side views that the single-view model neither observes nor generates. Without generated depth, and hence without CPP on the side-view cells, the three views add only 0.4 over the front view alone (App. C.4), so most of the gain arrives via CPP.

Hinge schedule. Both hinges are one-sided and vanish on every plan that keeps its margins, so late in training the balancing of Sec. 3.4 would concentrate their fixed gradient fraction on few plans that still violate them; we therefore anneal both hinges to zero by the middle of training. CPP penalizes deviation in either direction on every LiDAR-supported cell, so it does not vanish once plans are safe, and removing it from update 20,000 onward costs 1.2 HD-Score on HUGSIM (App. C.4).

Qualitative examples. Figure 3 shows both ablations on individual scenes: without CPP the plan drives into a parked van that the full model stops short of, and without the side views the plan cuts a left turn into an oncoming car that only the left camera shows at $t = 0$.

4.3 World-Model Quality

Metric depth. Scored against the LiDAR sweep of the same future frame and camera without scale alignment, front-view depth reaches AbsRel 0.175 at +2 s and 0.232 at +4 s (Table 3c), and 0.144 and 0.194 on the cells that CPP supervises. These values are below every reported future-depth result on nuScenes (GeoWAM [40], 0.245 and 0.297; Epona [66] with a DVGT [80] head, 0.263 and 0.310; VGGT-World [47], 0.329 and 0.357; Cosmos 3 [2] + DVGT, 0.376 and 0.422); the side views and the error structure are in App. C.5.

Video. The generated front video reaches FVD-9 111.3 against a recorded-versus-recorded floor of 91.5 at 600 clips per side, and 42.2 over all 1,200 clips (Table 14); reported NAVSIM values range from 142.6 (DrivingGPT [8]) to 32.7 (CoWorld-VLA [22]) under their own protocols. The generated video and depth also agree with the generated motion: the camera yaw recovered from the video deviates from it by a median of $0.80°$ over 4 s against a $0.29°$ floor on recorded clips, and the front and side depths agree to within $1.5\times$ the LiDAR floor (App. C.5).

5 Conclusion

We presented PHYSWAM, a world–action model that denoises multi-view video, metric depth and ego motion in one generator and couples the generated depth and motion through Coupled Point Projection (CPP), one loss on projected LiDAR geometry in metric space, alongside two geometric hinge losses and output-gradient balancing. The model plans strongly on NAVSIM, transfers zero-shot to closed-loop driving and generates accurate future depth. Removing CPP degrades both planning and depth performance. A self-supervised form of the coupling would let the model scale to driving data without LiDAR or dense labels, and consistency on rollouts away from the strictly recorded trajectory. Additionally, backbones and scales beyond the one studied here, are the relevant directions we find promising to explore (limitations in App. D).

AI Use Statement

AI models, including Claude and ChatGPT, were used as general-purpose assistants under the authors' supervision: for writing and polishing the paper, for implementing the method, and for running and monitoring the experiments. The authors hold responsibility for the content of the paper.

Ethics Statement

This work uses publicly available driving datasets and pretrained models under their respective licenses and involves no human subjects and no data collection or annotation beyond what those datasets provide. PHYSWAM is a research model evaluated on recorded data and in simulation; it is not validated for deployment on public roads, and its trajectories are not meant to control a vehicle without the safety systems such deployment requires.

Reproducibility Statement

PHYSWAM builds on open-source datasets (NAVSIM and its OpenScene source data, HUGSIM for closed-loop evaluation) and on the publicly released Cosmos 3 Nano checkpoint, which we fine-tune. We will release our code, training and evaluation scripts, and model checkpoints. The appendices give the full details needed to reproduce the results: the architecture and objective (App. A), data and label construction (App. B), and hyperparameters and evaluation protocols (App. C).

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[2] Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

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

[4] Daniel Bogdoll, Yitian Yang, Tim Joseph, Melih Yazgan, and J Marius Zollner. Muvo: A multimodal generative world model for autonomous driving with geometric representations. In *2025 IEEE Intelligent Vehicles Symposium (IV)*, pages 2243–2250. IEEE, 2025.

[5] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

[6] Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, et al. Pseudo-simulation for autonomous driving. *arXiv preprint arXiv:2506.04218*, 2025.

[7] Anthony Chen, Wenzhao Zheng, Yida Wang, Xueyang Zhang, Kun Zhan, Peng Jia, Kurt Keutzer, and Shanghang Zhang. Geodrive: 3d geometry-informed driving world model with precise action control. *arXiv preprint arXiv:2505.22421*, 2025.

[8] Yuntao Chen, Yuqi Wang, and Zhaoxiang Zhang. Drivinggpt: Unifying driving world modeling and planning with multi-modal autoregressive transformers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 26890–26900. IEEE, 2025.

[9] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018.

[10] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE transactions on pattern analysis and machine intelligence*, 45(11):12878–12895, 2022.

[11] OpenScene Contributors. Openscene: The largest up-to-date 3d occupancy prediction benchmark in autonomous driving. `https://github.com/OpenDriveLab/OpenScene`, 2023.

[12] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, et al. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *Advances in Neural Information Processing Systems*, 37:28706–28719, 2024.

[13] Renju Feng, Ning Xi, Duanfeng Chu, Rukang Wang, Zejian Deng, Anzheng Wang, Liping Lu, Jinxiang Wang, and Yanjun Huang. Artemis: Autoregressive end-to-end trajectory planning with mixture of experts for autonomous driving. *IEEE Robotics and Automation Letters*, 11 (1):226–233, 2025.

[14] Jiacheng Fu, Yibo Yuan, Meng Tian, Yue Li, Jiangtong Zhu, Jianhua Han, Yueyi Zhang, Jianwu Fang, Jianru Xue, Hang Xu, et al. 4d-wam: 4d consistent world modeling for autonomous driving. *arXiv preprint arXiv:2608.10107*, 2026.

[15] Ruiyuan Gao, Kai Chen, Enze Xie, Lanqing Hong, Zhenguo Li, Dit-Yan Yeung, and Qiang Xu. Magicdrive: Street view generation with diverse 3d geometry control. In *International Conference on Learning Representations*, volume 2024, pages 22841–22860, 2024.

[16] Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. Vista: A generalizable driving world model with high fidelity and versatile controllability. *Advances in Neural Information Processing Systems*, 37:91560–91596, 2024.

[17] Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4d world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.

[18] Yufeng Hong, Xiaotian Zhou, Yingyan Li, Xiangpo Zhou, Lin Liu, Yadan Luo, Shaoqing Xu, Lei Yang, and Ziying Song. Drivefuture: Future-aware latent world models for autonomous driving. *arXiv preprint arXiv:2605.09701*, 2026.

[19] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[21] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17853–17862. IEEE, 2023.

[22] Minqing Huang, Yujiao Xiang, Zihan Liang, Jiajie Huang, Jingqi Wang, Zhi Xu, Feiyang Tan, Hangning Zhou, Mu Yang, and Gong Che. Coworld-vla: Thinking in a multi-expert world model for autonomous driving. *arXiv preprint arXiv:2605.10426*, 2026.

[23] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pages 492–518. Springer, 1992.

[24] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8306–8316. IEEE, 2023.

[25] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, et al. Mapanything: Universal feed-forward metric 3d reconstruction; map-anything. github. io. In *2026 International Conference on 3D Vision (3DV)*, pages 499–509. IEEE, 2026.

[26] Bohan Li, Zhuang Ma, Dalong Du, Baorui Peng, Zhujin Liang, Zhenqiang Liu, Xianda Guo, Zheng Zhu, Chao Ma, Yueming Jin, et al. Omninwm: Omniscient driving navigation world models. *arXiv preprint arXiv:2510.18313*, 2025.

[27] Kailin Li, Zhenxin Li, Shiyi Lan, Yuan Xie, Zhizhong Zhang, Jiayi Liu, Zuxuan Wu, Zhiding Yu, and Jose M Alvarez. Hydra-mdp++: Advancing end-to-end driving via expert-guided hydra-distillation. *arXiv preprint arXiv:2503.12820*, 2025.

[28] Yingyan Li, Lue Fan, Jiawei He, Yuqi Wang, Yuntao Chen, Zhaoxiang Zhang, and Tieniu Tan. Enhancing end-to-end autonomous driving with latent world model. In *International Conference on Learning Representations*, volume 2025, pages 42942–42959, 2025.

[29] Yingyan Li, Yuqi Wang, Yang Liu, Jiawei He, Lue Fan, and Zhaoxiang Zhang. End-to-end driving with online trajectory evaluation via bev world model. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 27137–27146. IEEE, 2025.

[30] 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*, volume 2026, pages 7890–7911, 2026.

[31] Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, et al. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024.

[32] Dingkang Liang, Dingyuan Zhang, Xin Zhou, Sifan Tu, Tianrui Feng, Xiaofan Li, Yumeng Zhang, Mingyang Du, Xiao Tan, and Xiang Bai. Unifuture: A 4d driving world model for future generation and perception. *arXiv preprint arXiv:2503.13587*, 2025.

[33] Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024.

[34] Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, et al. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12037–12047. IEEE, 2025.

[35] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[36] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[37] Lin Liu, Ziying Song, Caiyan Jia, Hangjun Ye, Xiaoshuai Hao, Long Chen, et al. Driveworld-vla: Unified latent-space world modeling with vision-language-action for autonomous driving. *arXiv preprint arXiv:2602.06521*, 2026.

[38] 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*, pages 315–335. Springer, 2026.

[39] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[40] Yiren Lu, Xin Ye, Jiaming Liu, Philip Jacobson, Jin Yao, Yi-chung Chen, Liam Merino, Dhruva Dixith Kurra, Min Cai, Tom Lampo, et al. Geowam: Visual geometry world action models for autonomous driving. *arXiv preprint arXiv:2608.23486*, 2026.

[41] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 577–582. 2023.

[42] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7073–7083. IEEE, 2021.

[43] Lloyd Russell, Anthony Hu, Lorenzo Bertoni, George Fedoseev, Jamie Shotton, Elahe Arani, and Gianluca Corrado. Gaia-2: A controllable multi-view generative world model for autonomous driving. *arXiv preprint arXiv:2503.20523*, 2025.

[44] Zihao Sheng, Xin Ye, Jingru Luo, Sikai Chen, and Liu Ren. Explorevla: Dense world modeling and exploration for end-to-end autonomous driving. In *European Conference on Computer Vision*, pages 266–284. Springer, 2026.

[45] 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.

[46] Wenchao Sun, Xuewu Lin, Keyu Chen, Zixiang Pei, Xiang Li, Yining Shi, and Sifa Zheng. Sparsedrivev2: Scoring is all you need for end-to-end autonomous driving. In *European Conference on Computer Vision*, pages 446–463. Springer, 2026.

[47] Xiangyu Sun, Shijie Wang, Fengyi Zhang, Lin Liu, Caiyan Jia, Ziying Song, Zi Huang, and Yadan Luo. Vggt-world: Transforming vggt into an autoregressive geometry world model. In *European Conference on Computer Vision*, pages 382–400. Springer, 2026.

[48] 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.

[49] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5294–5306. IEEE, 2025.

[50] Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21486–21499, June 2026.

[51] Junli Wang, Zhihua Hua, Xueyi Liu, Zebin Xing, Wei Zhang, Kun Ma, Guang Chen, Hangjun Ye, Long Chen, and Qichao Zhang. Beyond imitation: Learning safe end-to-end autonomous driving from hard negatives. In *European Conference on Computer Vision*, pages 224–241. Springer, 2026.

[52] Linhan Wang, Zichong Yang, Chen Bai, Guoxiang Zhang, Xiaotong Liu, Xiaoyin Zheng, Xiao-Xiao Long, Chang-Tien Lu, and Cheng Lu. Drive-jepa: Video jepa meets multimodal trajectory distillation for end-to-end driving. *arXiv preprint arXiv:2601.22032*, 2026.

[53] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[54] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20697–20709. IEEE, 2024.

[55] Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. Drivedreamer: Towards real-world-drive world models for autonomous driving. In *European conference on computer vision*, pages 55–72. Springer, 2024.

[56] Yuqi Wang, Jiawei He, Lue Fan, Hongxin Li, Yuntao Chen, and Zhaoxiang Zhang. Driving into the future: Multiview visual forecasting and planning with world model for autonomous driving. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14749–14759. IEEE, 2024.

[57] Haoyu Wu, Diankun Wu, Tianyu He, Junliang Guo, Yang Ye, Yueqi Duan, and Jiang Bian. Geometry forcing: Marrying video diffusion and 3d representation for consistent world modeling. In *International Conference on Learning Representations*, volume 2026, pages 119186–119209, 2026.

[58] Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, Kun Ma, et al. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. In *International Conference on Learning Representations*, volume 2026, pages 157518–157556, 2026.

[59] Jiawei Xu, Zhizhou Zhong, Zhijian Shu, Mingkai Jia, Mingxiao Li, Jia-Wang Bian, Qian Zhang, Kaicheng Zhang, Jin Xie, Jian Yang, et al. Eponav2: Driving world model with comprehensive future reasoning. *arXiv preprint arXiv:2605.14696*, 2026.

[60] Jiazhi Yang, Kashyap Chitta, Shenyuan Gao, Long Chen, Yuqian Shao, Xiaosong Jia, Hongyang Li, Andreas Geiger, Xiangyu Yue, and Li Chen. Resim: Reliable world simulation for autonomous driving. *Advances in Neural Information Processing Systems*, 38:167710–167741, 2026.

[61] Wenhao Yao, Zhenxin Li, Shiyi Lan, Zi Wang, Xinglong Sun, Jose M Alvarez, and Zuxuan Wu. Drivesuprim: Towards precise trajectory selection for end-to-end planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 11910–11918, 2026.

[62] Li Yaokun, Shuaixian Wang, Mantang Guo, Jiehui Huang, Taojun Ding, Mu Hu, Kaixuan Wang, Shaojie Shen, and Guang Tan. Recamdriving: Lidar-free camera-controlled video synthesis for novel trajectories. In *European Conference on Computer Vision*, pages 301–319. Springer, 2026.

[63] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

[64] Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K Barral, Raia Hadsell, et al. Efficiently reconstructing dynamic scenes one d4rt at a time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7382–7392, 2026.

[65] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Forrester Cole, Deqing Sun, Ming-Hsuan Yang, et al. Monst3r: A simple approach for estimating geometry in the presence of motion. In *International Conference on Learning Representations*, volume 2025, pages 82863–82886, 2025.

[66] 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 *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 27220–27230. IEEE, 2025.

[67] Songyan Zhang, Jinyuan Tian, Hanbing Li, Daqi Liu, Hao Chen, Wenhui Huang, Fang Li, Guang Chen, Hangjun Ye, Long Chen, et al. Geoworldad: Geometry world action model for autonomous driving. *arXiv preprint arXiv:2607.17521*, 2026.

[68] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=hrkmlPhp1u`.

[69] Zhida Zhao, Talas Fu, Yifan Wang, Lijun Wang, and Huchuan Lu. From forecasting to planning: Policy world model for collaborative state-action prediction. *Advances in Neural Information Processing Systems*, 38:134585–134611, 2026.

[70] Zongchuang Zhao, Xin Zhou, Tianyang Xu, Zhengyang Sun, Kaixuan Zhou, Honglin Li, Dingkang Liang, and Xiang Bai. Simwam: A simple world action model for end-to-end autonomous driving. *arXiv preprint arXiv:2608.07468*, 2026.

[71] Wenzhao Zheng, Weiliang Chen, Yuanhui Huang, Borui Zhang, Yueqi Duan, and Jiwen Lu. Occworld: Learning a 3d occupancy world model for autonomous driving. In *European conference on computer vision*, pages 55–72. Springer, 2024.

[72] Yupeng Zheng, Pengxuan Yang, Zebin Xing, Qichao Zhang, Yuhang Zheng, Yinfeng Gao, Pengfei Li, Teng Zhang, Zhongpu Xia, Peng Jia, et al. World4drive: End-to-end autonomous driving via intention-aware physical latent world model. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 28632–28642. IEEE, 2025.

[73] 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. *arXiv preprint arXiv:2608.19085*, 2026.

[74] Hongyu Zhou, Longzhong Lin, Jiabao Wang, Yichong Lu, Dongfeng Bai, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugsim: A real-time, photo-realistic and closed-loop simulator for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[75] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pages 6612–6619. IEEE, 2017.

[76] Yang Zhou, Xiaofeng Wang, Hao Shao, Letian Wang, Guosheng Zhao, Jiangnan Shao, Jiagang Zhu, Tingdong Yu, Zheng Zhu, Guan Huang, et al. Drivedreamer-policy: A geometry-grounded world-action model for unified generation and planning. *arXiv preprint arXiv:2604.01765*, 2026.

[77] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5738–5746. IEEE, 2019.

[78] Zewei Zhou, Tianhui Cai, Seth Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *Advances in Neural Information Processing Systems*, 38:27920–27956, 2026.

[79] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

[80] Sicheng Zuo, Zixun Xie, Wenzhao Zheng, Shaoqing Xu, Fang Li, Shengyin Jiang, Long Chen, Zhi-Xin Yang, and Jiwen Lu. Dvgt: Driving visual geometry transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14658–14668, 2026.

[81] Sicheng Zuo, Zixun Xie, Wenzhao Zheng, Shaoqing Xu, Fang Li, Hanbing Li, Long Chen, Zhi-Xin Yang, and Jiwen Lu. Dvgt-2: Vision-geometry-action model for autonomous driving at scale. *arXiv preprint arXiv:2604.00813*, 2026.

A Additional Method Details

This appendix follows the order of Sec. 3: the representations on which the losses act (Sec. A.1), the depth head and the gradients of CPP (Sec. A.2), the distance functions of the hinge losses (Sec. A.3), output-gradient balancing and the complete update (Sec. A.4), and inference (Sec. A.5). Notation follows the main text: t indexes scene frames, j latent frames, and σ the noise level; $\mathbf{x}$ denotes VAE-grid or pose representations and $\mathbf{z}$ transformer tokens.

A.1 Architecture and Representations

Pretrained architecture and adaptation. PHYSWAM builds on the Cosmos 3 Mixture-of-Transformers architecture [2; 33], based on Qwen3-VL [3]. Within each layer, understanding (text) tokens and generation tokens have separate normalization, query/key/value projections, output projections, and feed-forward parameters, and share one attention operation. Text queries attend causally to text; generation queries attend bidirectionally to all text, visual, and action tokens of the same example. Text and generated content therefore interact in every layer rather than through a completed prediction passed from one model to another. All RGB and depth tokens, including the clean RGB context, use the generation pathway. We adapt the generation pathway with its input projections, output projections, and embeddings, and keep the text pathway, the Wan video VAE [48], and the depth head of Sec. A.2 fixed. RGB and depth share the VAE and the visual projections; the depth embedding $\mathbf{e}_D$ distinguishes the two modalities and the ray embedding identifies their viewing geometry.

Token layout. Table 4 follows a visual stream $i = (v, m)$ from image to velocity. With $\bar{h}_v = \lceil h_v/p \rceil$ and $\bar{w}_v = \lceil w_v/p \rceil$, the stream has $N_i = J\bar{h}_v\bar{w}_v$ tokens, of which N_i^{gen} lie at generated frames. Padding introduced for the VAE stride and for patchification is removed before any loss or depth estimate is formed, so all quantities below refer to the content grid.

Table 4: **Visual representation pathway** for one stream $i = (v, m)$. Shapes omit the minibatch dimension. Noise and flow targets are defined on the VAE grid, before patchification; each spatial patch becomes one token.

Quantity	Shape	Operation
RGB or encoded depth	$(T+1) \times H_v \times W_v \times 3$	Input to the shared frozen VAE.
$\mathbf{x}_i$	$J \times h_v \times w_v \times c_\ell$	VAE grid, indexed spatially by u.
$\mathcal{P}(\mathbf{x}_i^\sigma)$	$N_i \times p^2 c_\ell$	Flattened patches of the noisy grid, indexed by q.
$\mathbf{z}_i$	$N_i \times d_h$	Projected patches with ray, modality, and noise embeddings.
$\mathcal{O}_V(\mathbf{z}_i^{\text{out}})$	$N_i^{\text{gen}} \times p^2 c_\ell$	Patch velocities at generated positions.
$\widehat{\mathbf{v}}_i$	$J \times h_v \times w_v \times c_\ell$	Unpatchified velocity field; conditioning frames are zero-filled and not predicted.

Metric depth encoding. The VAE receives depth as an image. For fixed bounds $d_{\min}, d_{\max}$, with $\ell_{\min} = \log(1 + d_{\min})$ and $\ell_{\max} = \log(1 + d_{\max})$, the scalar encoding is

$$\mathcal{C}(d) = 2\frac{\ell_{\max} - \log(1 + \text{clip}(d, d_{\min}, d_{\max}))}{\ell_{\max} - \ell_{\min}} - 1, \tag{9}$$

so that nearer surfaces are brighter and the full range maps onto $[-1, 1]$. The scalar is quantized to the image format, replicated across the three channels, and normalized to the VAE input range. Decoding averages the reconstructed channels and inverts the continuous map,

$$\mathcal{C}^{-1}(y) = \exp\left(\ell_{\max} - \frac{\text{clip}(y, -1, 1) + 1}{2}(\ell_{\max} - \ell_{\min})\right) - 1. \tag{10}$$

Because the bounds are shared by all scenes, a decoded depth is metric without a per-scene scale; quantization and VAE reconstruction add approximation error but leave this convention intact.

Camera geometry at two resolutions. With VAE spatial stride s_x, the image-plane centres of latent cell $u = (u_h, u_w)$ and of token patch $q = (q_h, q_w)$ are

$$\boldsymbol{\xi}_v^{\text{cell}}(u) = s_x\big(u_w + \tfrac{1}{2},\, u_h + \tfrac{1}{2}\big), \qquad \boldsymbol{\xi}_v(q) = s_x p\big(q_w + \tfrac{1}{2},\, q_h + \tfrac{1}{2}\big), \tag{11}$$

in the coordinates of the resized image, with calibration scaled accordingly; the rays account for lens distortion. For the ray embedding, the unit camera-frame ray through $\boldsymbol{\xi}_v(q)$ is rotated by $\text{rot}(\mathbf{G}_v)$ into the ego frame and paired with the camera centre $\text{trans}(\mathbf{G}_v)$ to form the Plücker coordinates $\mathbf{r}_v(q)$ of Eq. (3). CPP uses the camera-frame ray $\mathbf{b}_v(u)$ through $\boldsymbol{\xi}_v^{\text{cell}}(u)$. The embedding and the unprojection thus share one camera model and differ only in resolution.

Temporal coordinates. mRoPE [53; 2] assigns every token a (time, height, width) position. The spatial offsets (b_v^h, b_v^w) in Eq. (4) are in token-grid units and place the left, front, and right grids side by side; camera geometry enters only through the ray embedding. Along time, successive visual latents advance by $\alpha s_t/f$ and successive actions by α/f, so action $t = \kappa(j)$ and latent frame j share a temporal coordinate although their sequence positions differ. Every visual stream starts at the common offset τ_0 after the text, and text tokens use their causal index on all three axes. Scene time is thereby encoded in the attention geometry, while the noise level of generated tokens is carried separately by $\mathbf{e}_\sigma$.

Stream losses. The current RGB latent and the motion history are clean; all depth latents and all future RGB and motion entries are generated, with one σ per example and independent Gaussian entries across its streams. For stream i, let $\mathcal{I}_i$ be the scalar entries of its retained (unpadded) representation and $M_i(e) \in \{0, 1\}$ indicate a generated entry. The stream loss in Eq. (5) is

$$\mathcal{L}_{\mathrm{FM}}^{(i)} = \frac{1}{|\mathcal{I}_i|} \sum_{e \in \mathcal{I}_i} M_i(e) \big(\widehat{\mathbf{v}}_i(e) - \mathbf{v}_i^\star(e)\big)^2. \tag{12}$$

Conditioning entries contribute zero error but remain in the denominator, so the normalization of a stream does not depend on how many of its entries are generated. For the action stream, $\mathcal{I}_A$ comprises the history row and the future rows over the nine pose coordinates. Each visual stream is reduced to its mean before the weights $\omega_{v,m}$ are applied, so a front view with many tokens and a side view with few enter the visual loss on the same footing; the expectation in Eq. (5) then averages over examples.

A.2 Depth Head and Coupled Point Projection

Depth head. The head Π_ϕ of Sec. 3.3 stacks three 3×3 convolutions with channel widths $c_\ell \to c_\Pi \to c_\Pi \to 1$ and GELU after the first two, at stride one with symmetric padding, so the VAE grid is preserved. It acts on each latent frame independently, is shared by the three views, and takes the estimated clean latent $\widehat{\mathbf{x}}_{v,D}$ after unpatchification, in the normalization of the generative model.

We fit ϕ once, before world-model training, on clean latents $\mathbf{x}_{v,D}$ and on estimated-clean latents $\widehat{\mathbf{x}}_{v,D}$ computed from the velocity predictions of a preliminary model with the same architecture and representation, at noise levels drawn from the training distribution and weighted by $w(\sigma)$, so that the head is most accurate where CPP carries most weight. The target of a latent cell with LiDAR depths $\{d_r\}_{r=1}^{n_u}$ is their geometric mean $d^L = \exp(n_u^{-1} \sum_r \log d_r)$, regressed as $\log(1 + d^L)$. Let $\mathcal{F}_v$ collect the (latent frame, valid cell, target) triples $(\widetilde{\mathbf{x}}, u, d^L)$ of view v in a fitting minibatch. The fitting loss is

$$\mathcal{L}_{\mathrm{head}}(\phi) = \sum_{v \in \mathcal{V}} \frac{1}{|\mathcal{F}_v|} \sum_{(\widetilde{\mathbf{x}}, u, d^L) \in \mathcal{F}_v} \rho_\beta\big(\big|[\Pi_\phi(\widetilde{\mathbf{x}})](u) - \log(1 + d^L)\big|\big), \tag{13}$$

where ρ_β is the Huber penalty of Sec. 3.3 with transition β in log-depth units. After fitting, ϕ is fixed: CPP gradients pass through Π_ϕ into the generated latents, and the head cannot absorb geometric error by adapting to the generator.

Correspondence and anchors. CPP compares two 3D points per LiDAR-supported cell, the generated $\widehat{\mathbf{P}}_{v,t}(u)$ and the measured $\mathbf{P}_{v,t}^\star(u)$ of Eq. (6). Both lie on the ray $\mathbf{b}_v(u)$ and are placed in $\mathcal{C}_0$ by $\mathbf{E}_v$ followed by the generated chain $\widehat{\mathbf{T}}_t$ or the recorded pose $\mathbf{T}_t^\star$, so correspondence is given by the index (v, t, u) and no point-cloud registration is needed. The anchors $\mathcal{A} \subseteq \{\kappa(j) : j > 0\}$ are future frames: at $t = 0$ the transform is the identity and the comparison would supervise depth alone. A cell is valid when it holds at least $n_{\min}$ LiDAR returns, its pooled depth is finite and lies in $(0, d_{\max}]$, and its predicted depth is finite; invalid cells receive zero weight and zero gradient.

Reduction. For example b and view v, let $n_{b,v}$ count the valid cells over all anchors and $\mathcal{L}_{\mathrm{cpp}}^{(b,v)}$ be their mean Huber error. With $\mathcal{B}_+ = \{b : \sum_v n_{b,v} > 0\}$, the noise-weighted minibatch loss is

$$\overline{\mathcal{L}}_{\mathrm{cpp}} = \frac{1}{|\mathcal{B}_+|} \sum_{b \in \mathcal{B}_+} w(\sigma_b) \frac{\sum_v n_{b,v} \mathcal{L}_{\mathrm{cpp}}^{(b,v)}}{\sum_v n_{b,v}}. \tag{14}$$

Each example averages over all of its valid correspondences, so a densely observed view carries proportionally more weight than a sparsely observed one, and examples without LiDAR support are left out of the mean rather than diluting it.

Gradients to depth and motion. Let $\mathbf{M} = \widehat{\mathbf{T}}_t \mathbf{E}_v$ with rotation $\mathbf{R}_\mathbf{M}$, let $\mathbf{r} = \widehat{\mathbf{P}}_{v,t}(u) - \mathbf{P}^\star_{v,t}(u)$ be the residual with $n = \|\mathbf{r}\|_2$, and let $\mathbf{g} = \rho'_\delta(n)\,\mathbf{r}/n$ with $\rho'_\delta(n) = \min(n/\delta, 1)$. For one correspondence, the Huber term $\rho_\delta(n)$ has the gradients

$$\frac{\partial \rho_\delta}{\partial \widehat{d}} = \mathbf{g}^\top \mathbf{R}_\mathbf{M} \frac{\mathbf{b}_v(u)}{b_{v,z}(u)}, \qquad \frac{\partial \rho_\delta}{\partial \widehat{\mathbf{t}}_t} = \mathbf{g}, \qquad \frac{\partial \rho_\delta}{\partial \boldsymbol{\theta}_t} = \big(\widehat{\mathbf{P}}_{v,t}(u) - \widehat{\mathbf{t}}_t\big) \times \mathbf{g}, \tag{15}$$

where $\widehat{d} = \widehat{d}^{\,v}_{j(t)}(u)$, $\widehat{\mathbf{t}}_t$ is the translation of $\widehat{\mathbf{T}}_t$, and $\boldsymbol{\theta}_t$ is a small rotation of the generated camera about its own centre $\widehat{\mathbf{t}}_t$, expressed in $\mathcal{C}_0$. The generated depth is thus corrected along its transformed viewing ray, by the component of the residual in that direction, while the generated pose receives the residuals of all cells at the anchor as a net force on its translation and a net torque about the camera centre on its rotation. The pose gradient reaches every relative motion up to t through the composition $\widehat{\mathbf{T}}_t = \boldsymbol{\Delta}(\widehat{\mathbf{a}}_1) \cdots \boldsymbol{\Delta}(\widehat{\mathbf{a}}_t)$, and the depth gradient reaches the latent $\widehat{\mathbf{x}}_{v,D}$ through Π_ϕ; both then reach the velocity outputs through Eq. (2), with the factor $-\sigma$. One residual vector drives both branches, which is the coupling of Sec. 3.3 in explicit form.

Relation to decoupled supervision. Write $\mathbf{p}(d) = \mathbf{E}_v\, d\, \mathbf{b}_v(u)/b_{v,z}(u)$ for the point at depth d on the ray of cell u, so that $\widehat{\mathbf{P}}_{v,t}(u) = \widehat{\mathbf{T}}_t\, \mathbf{p}(\widehat{d})$ and $\mathbf{P}^\star_{v,t}(u) = \mathbf{T}^\star_t\, \mathbf{p}(d^L)$. The residual of one correspondence, $\mathbf{r} = \widehat{\mathbf{P}}_{v,t}(u) - \mathbf{P}^\star_{v,t}(u)$, is the vector whose Huber norm CPP penalizes. It splits exactly into three parts,

$$\mathbf{r} = \underbrace{\mathbf{T}^\star_t\, \mathbf{p}(\widehat{d}) - \mathbf{T}^\star_t\, \mathbf{p}(d^L)}_{\mathbf{r}_D} + \underbrace{\widehat{\mathbf{T}}_t\, \mathbf{p}(d^L) - \mathbf{T}^\star_t\, \mathbf{p}(d^L)}_{\mathbf{r}_T} + \underbrace{\big(\widehat{\mathbf{T}}_t - \mathbf{T}^\star_t\big)\big(\mathbf{p}(\widehat{d}) - \mathbf{p}(d^L)\big)}_{\mathbf{r}_{DT}}. \tag{16}$$

The first part, $\mathbf{r}_D = (\widehat{d} - d^L)\, \mathbf{R}^\star_t \mathbf{R}_{\mathbf{E}_v} \mathbf{b}_v(u)/b_{v,z}(u)$, is the depth error carried along the recorded transformed ray; it depends on the generated depth only. The second, $\mathbf{r}_T$, applies the generated pose to the measured point and compares with the recorded pose; it depends on the generated motion only. The third, $\mathbf{r}_{DT}$, is the product of the two errors.

A decoupled objective would penalize the two errors separately, as $\rho_\delta(\|\mathbf{r}_D\|) + \rho_\delta(\|\mathbf{r}_T\|)$. Neither term is new to the objective. The first is a Huber loss on the metric depth error at the LiDAR-supported cells, up to the ray obliquity $\|\mathbf{b}_v(u)\|/b_{v,z}(u)$. The depth stream already receives this supervision at every cell through its flow-matching target, a dense label anchored to the same LiDAR sweep (App. B.2). The second is a loss on the composed pose $\widehat{\mathbf{T}}_t$, evaluated on the measured points: the translation error enters directly, and the rotation error is scaled by the distance of the point. The hinges (App. A.3) are likewise a loss on the composed clean-estimate positions, and the action flow-matching target supervises the per-step actions themselves.

CPP applies ρ_δ to the sum instead, and this changes the gradients. With $\mathbf{g}$ the Huber-scaled residual of Eq. (15), the depth gradient $\mathbf{g}^\top \mathbf{R}_\mathbf{M} \mathbf{b}_v(u)/b_{v,z}(u)$ contains the components of $\mathbf{r}_T$ and $\mathbf{r}_{DT}$ along the ray, so the depth is moved by the motion error. The pose gradients $\mathbf{g}$ and $(\widehat{\mathbf{P}}_{v,t}(u) - \widehat{\mathbf{t}}_t) \times \mathbf{g}$ contain $\mathbf{r}_D$, so the motion is moved by the depth error. The two branches share one residual and one Huber scale: each is corrected toward agreement with the other's current prediction as well as with the measurement, and a cell whose joint error is large drives both. The ablation of Sec. 4.2 measures what this adds: CPP gains 1.0 PDMS over flow matching with the hinges, which already supply the composed-position term, and 2.1 over flow matching alone.

A.3 Geometric Hinge Losses

CPP supervises the generated depth–motion pair; the two hinges supervise waypoint positions through the generated motion alone, independently of generated depth. Both hinges act on the planar waypoints $\widehat{\mathbf{p}}_t = \pi_{xz}(\mathrm{trans}(\widehat{\mathbf{T}}_t))$, $t = 1, \ldots, T$, in the (x, z) plane of $\mathcal{C}_0$, which serves as the ground plane: x points right and z forward from the front camera at the current time, and height is discarded. Labels observed at a future time t are placed in this plane with the recorded pose $\mathbf{T}^\star_t$, as the measured points of CPP are, so waypoints and labels are compared in one frame. Both margins apply to positions rather than heading-dependent vehicle footprints, and the obstacle and map targets supply training supervision only.

Obstacle labels. For each future time t, the static set $\mathcal{S}_t \subset \mathbb{R}^2$ holds the planar positions of LiDAR returns from the sweep at t that belong to unannotated structure at vehicle height, such as curbs, poles, and barriers, and the box set $\mathcal{O}_t$ holds the footprints of the objects annotated at t, each an oriented rectangle $B = (\mathbf{c}_B, \mathbf{h}_B, \psi_B)$ with centre $\mathbf{c}_B \in \mathbb{R}^2$, half-extents $\mathbf{h}_B = (\frac{1}{2}L_B, \frac{1}{2}W_B)$ along

its length and width, and yaw ψ_B. Neither set contains the ego vehicle; App. B describes their construction.

Signed distance to an oriented rectangle. For a position $\mathbf{p}$, let $\mathbf{q} = \mathbf{R}_2(-\psi_B)(\mathbf{p} - \mathbf{c}_B)$ be its coordinates along the length and width axes of B, where $\mathbf{R}_2(\psi)$ is the planar rotation by ψ and ψ_B is the angle from the x axis to the length axis, and let $\boldsymbol{\zeta}_B(\mathbf{p}) = |\mathbf{q}| - \mathbf{h}_B = (\zeta_{B,1}, \zeta_{B,2})$ be the elementwise excess of $|\mathbf{q}|$ over the half-extents. The signed distance to the boundary of B is

$$\mathrm{sd}(\mathbf{p}, B) = \big\| \mathrm{ReLU}(\boldsymbol{\zeta}_B(\mathbf{p})) \big\|_2 + \min\big(\max(\zeta_{B,1}(\mathbf{p}), \zeta_{B,2}(\mathbf{p})),\, 0 \big). \tag{17}$$

Outside the rectangle at least one component of $\boldsymbol{\zeta}_B$ is positive, the second term vanishes, and the first is the Euclidean distance to the nearest edge or corner. Inside, both components are negative, the first term vanishes, and the second is minus the distance to the nearest edge. The function is continuous across the boundary, and its gradient has unit norm almost everywhere and points away from the rectangle on both sides of the boundary. An unsigned distance would instead grow towards the centre of the footprint, and the hinge would drive a penetrating waypoint deeper into it.

Obstacle clearance. For a position $\mathbf{p}$, the clearance at time t is

$$d_t(\mathbf{p}) = \min\left\{ \min_{\mathbf{s} \in \mathcal{S}_t} \|\mathbf{p} - \mathbf{s}\|_2,\ \min_{B \in \mathcal{O}_t} \mathrm{sd}(\mathbf{p}, B) \right\}, \tag{18}$$

where an empty set does not contribute and $d_t \equiv +\infty$ when $\mathcal{S}_t$ and $\mathcal{O}_t$ are both empty, so that the term vanishes; time-indexed annotations follow other vehicles' recorded locations. The recorded clearance $d_t^\star = d_t(\mathbf{p}_t^\star)$ evaluates it at the recorded waypoint $\mathbf{p}_t^\star = \pi_{xz}(\mathrm{trans}(\mathbf{T}_t^\star))$ and is negative when $\mathbf{p}_t^\star$ lies inside a footprint. Capping the margin at $m_t = \min(m_{\mathrm{obs}}, d_t^\star)$ makes the recorded trajectory incur zero loss at every t: in free space the hinge requires the nominal margin, and where the recorded path passed closer, only the clearance that path kept. For the nominal margin $m_{\mathrm{obs}} > 0$, the loss is

$$\mathcal{L}_{\mathrm{obs}} = \frac{1}{T} \sum_{t=1}^{T} \big[\mathrm{ReLU}\big(m_t - d_t(\widehat{\mathbf{p}}_t)\big) \big]^2, \tag{19}$$

with $\mathrm{ReLU}(x) = \max(x, 0)$, so only clearance below m_t is penalized. The gradient of the penalty $[\mathrm{ReLU}(m_t - d_t(\widehat{\mathbf{p}}_t))]^2$ with respect to $\widehat{\mathbf{p}}_t$ is $-2\,\mathrm{ReLU}(m_t - d_t(\widehat{\mathbf{p}}_t))\,\nabla d_t(\widehat{\mathbf{p}}_t)$, where $\nabla d_t(\widehat{\mathbf{p}}_t)$ is the unit vector to $\widehat{\mathbf{p}}_t$ from the nearest obstacle point (a static return, or the closest boundary point of a footprint, which near a corner is the corner itself) and, inside a footprint, the outward normal of its nearest edge. A violating waypoint is thus moved directly away from the nearest obstacle with a force proportional to the deficit, and the gradient vanishes once the margin is met.

Drivable-area label. The drivable region $\mathcal{R}$ is the union of the map polygons on which driving is permitted (road blocks, intersections, and car-park areas). Its signed distance field is stored on a raster $\mathbf{S} \in \mathbb{R}^{N_z \times N_x}$ of square cells with spacing r_S, covering $[x_{\min}, x_{\min} + r_S N_x) \times [z_{\min}, z_{\min} + r_S N_z)$ in the ground plane of $\mathcal{C}_0$, a region around the current camera position that extends farther forward than backward. The entry S_{i_z, i_x} is the Euclidean distance in metres from the cell centre $\big(x_{\min} + (i_x + \frac{1}{2}) r_S,\ z_{\min} + (i_z + \frac{1}{2}) r_S\big)$ to the boundary of $\mathcal{R}$, positive inside $\mathcal{R}$ and negative outside.

Drivable-area clearance. With $S(\mathbf{p})$ the signed distance to the drivable boundary, positive inside the drivable region, and the required interior clearance $m_{\mathrm{drv}} > 0$,

$$\mathcal{L}_{\mathrm{drv}} = \frac{1}{T} \sum_{t=1}^{T} \big[\mathrm{ReLU}\big(m_{\mathrm{drv}} - S(\widehat{\mathbf{p}}_t)\big) \big]^2, \tag{20}$$

which penalizes waypoints near or outside the drivable boundary and vanishes beyond the required interior margin. The value $S(\mathbf{p})$ is the bilinear interpolation of the raster at $\mathbf{p}$. The continuous cell indices

$$\iota_x(\mathbf{p}) = \frac{p_x - x_{\min}}{r_S} - \tfrac{1}{2}, \qquad \iota_z(\mathbf{p}) = \frac{p_z - z_{\min}}{r_S} - \tfrac{1}{2} \tag{21}$$

take integer values exactly at cell centres. With $i_x = \lfloor \iota_x \rfloor$, $a_x = \iota_x - i_x$, and likewise i_z, a_z,

$$S(\mathbf{p}) = (1-a_z)(1-a_x)\, S_{i_z, i_x} + (1-a_z)\, a_x\, S_{i_z, i_x+1} + a_z (1-a_x)\, S_{i_z+1, i_x} + a_z a_x\, S_{i_z+1, i_x+1}, \tag{22}$$

where indices outside the raster are clamped to its border, so a waypoint beyond the stored region takes the value of the nearest border cell. $S(\mathbf{p})$ is continuous and piecewise bilinear, and its gradient, obtained from the interpolation weights, approximates the unit inward normal of the drivable

boundary. The drivable penalty has the gradient of the obstacle case with S and m_{drv} in place of d_t and m_t, so a violating waypoint moves along ∇S, towards the interior; beyond the raster, the clamped coordinate receives no gradient.

Reduction. Each hinge averages its squared penalty over the T future waypoints of an example, is weighted by $w(\sigma)$, and is averaged over all examples of the minibatch; unlike CPP, every example contributes, since the labels are always available. Although only positions are penalized, $\widehat{\mathbf{p}}_t$ is the endpoint of the composed chain and depends on the translations of $\widehat{\mathbf{a}}_1, \dots, \widehat{\mathbf{a}}_t$ and the rotations of $\widehat{\mathbf{a}}_1, \dots, \widehat{\mathbf{a}}_{t-1}$; a heading error in $\widehat{\mathbf{a}}_{t'}$ is therefore corrected through every later waypoint it displaces. As for CPP, the gradients reach the action velocity output through Eq. (2) with the factor $-\sigma$.

A.4 Output-Gradient Balancing

The geometric losses differ from the flow-matching losses in units and sparsity, and their raw gradients at the velocity outputs can differ from the flow gradients by orders of magnitude. Fixed weights would have to follow these magnitudes as training changes them; the balancing of Sec. 3.4 instead recomputes the coefficients on every update from the gradients themselves. For target fractions η_k, each active geometric term contributes an η_k fraction of the action flow-gradient norm at the action output, and a gain γ on CPP's depth branch imposes the analogous ratio at the depth output; all gradients are measured before scaling, sg holds the coefficients fixed during backpropagation, and zero-denominator ratios are set to zero.

Gradients at the outputs. On the current minibatch, let the unweighted gradients, with the depth-side gain γ held at one, be

$$\begin{aligned} \mathbf{g}_A &= \nabla_{\widehat{\mathbf{v}}_A} \mathcal{L}_{\mathrm{FM},A}, \qquad \mathbf{g}_k^A = \nabla_{\widehat{\mathbf{v}}_A} \overline{\mathcal{L}}_k, \\ \mathbf{g}_D &= \nabla_{\widehat{\mathbf{v}}_D} \mathcal{L}_{\mathrm{FM},V}, \quad \mathbf{g}_{\mathrm{cpp}}^D = \nabla_{\widehat{\mathbf{v}}_D} \overline{\mathcal{L}}_{\mathrm{cpp}}, \end{aligned} \tag{23}$$

where $k \in \mathcal{K}$ and $\widehat{\mathbf{v}}_D$ collects the depth velocity outputs of all views, so that the visual flow gradient is restricted to depth for the depth-side comparison. Each vector concatenates the representation-space outputs of the whole minibatch; transformer hidden states are not involved. With the norms $\nu_A = \|\mathbf{g}_A\|_2$, $\nu_k^A = \|\mathbf{g}_k^A\|_2$, $\nu_D = \|\mathbf{g}_D\|_2$, and $\nu_{\mathrm{cpp}}^D = \|\mathbf{g}_{\mathrm{cpp}}^D\|_2$, the action coefficients read

$$s_k = \mathrm{sg}\left(\eta_k \frac{\nu_A}{\nu_k^A}\right), \tag{24}$$

and in the final backward pass the action output receives

$$\nabla_{\widehat{\mathbf{v}}_A} \mathcal{L}_{\mathrm{total}} = \lambda_A \Big(\mathbf{g}_A + \sum_{k \in \mathcal{K}} s_k \mathbf{g}_k^A\Big), \qquad \|\lambda_A s_k \mathbf{g}_k^A\|_2 = \eta_k \|\lambda_A \mathbf{g}_A\|_2 \tag{25}$$

for every active term. Each geometric term pushes the action output with η_k times the norm of the action flow push; because s_k is a constant in the backward pass, the identity holds exactly, at minibatch level, on the update for which s_k is computed.

Depth-side gain. CPP also reaches the depth output, and s_{cpp} alone does not balance it there. Substituting s_{cpp}, the ratio it would induce at the depth output is

$$\frac{\|\lambda_A s_{\mathrm{cpp}} \mathbf{g}_{\mathrm{cpp}}^D\|_2}{\|\lambda_V \mathbf{g}_D\|_2} = \eta_{\mathrm{cpp}} \frac{\lambda_A \nu_A}{\lambda_V \nu_D} \frac{\nu_{\mathrm{cpp}}^D}{\nu_{\mathrm{cpp}}^A}, \tag{26}$$

which equals η_{cpp} only when CPP divides its gradient between the depth and action outputs in the same proportion as the weighted flow losses divide theirs; nothing in the objective ties the two proportions. A second coefficient is therefore applied on the depth branch of CPP alone, through the operator

$$\mathcal{G}_\gamma(\widehat{d}) = \mathrm{sg}(\widehat{d}) + \gamma\big(\widehat{d} - \mathrm{sg}(\widehat{d})\big), \tag{27}$$

which returns $\widehat{d}$ unchanged in the forward pass and multiplies the gradient flowing back through it by γ; the value of CPP, its motion branch, and the flow-matching losses are unaffected. With the gain, the depth output receives

$$\nabla_{\widehat{\mathbf{v}}_D} \mathcal{L}_{\mathrm{total}} = \lambda_V \mathbf{g}_D + \lambda_A s_{\mathrm{cpp}} \gamma\, \mathbf{g}_{\mathrm{cpp}}^D, \qquad \|\lambda_A s_{\mathrm{cpp}} \gamma\, \mathbf{g}_{\mathrm{cpp}}^D\|_2 = \eta_{\mathrm{cpp}} \|\lambda_V \mathbf{g}_D\|_2, \tag{28}$$

and solving the identity for the gain gives

$$\gamma = \mathrm{sg}\left(\frac{\eta_{\mathrm{cpp}} \lambda_V \|\nabla_{\widehat{\mathbf{v}}_D} \mathcal{L}_{\mathrm{FM},V}\|_2}{\lambda_A s_{\mathrm{cpp}} \|\nabla_{\widehat{\mathbf{v}}_D} \overline{\mathcal{L}}_{\mathrm{cpp}}\|_2}\right), \tag{29}$$

Algorithm 1 One PHYSWAM training update with output-gradient balancing

Require: minibatch of examples with images $\mathbf{I}^v_{0:T}$, depth $\mathbf{D}^v_{0:T}$, motion $\mathbf{a}_{0:T}$, context c, pooled LiDAR depths, recorded poses $\mathbf{T}^\star_t$, obstacle sets $\mathcal{S}_t, \mathcal{O}_t$, and raster $\mathbf{S}$; parameters θ; fixed $\mathcal{E}$ and Π_ϕ; weights λ_V, λ_A; target fractions η_k

$\mathbf{x}_{v,I} \leftarrow \mathcal{E}(\mathbf{I}^v_{0:T})$, $\mathbf{x}_{v,D} \leftarrow \mathcal{E}(\mathcal{C}(\mathbf{D}^v_{0:T}))$, $\mathbf{x}_A \leftarrow \mathbf{a}_{0:T}$ ▷ clean representations

draw σ_b per example and $\boldsymbol{\epsilon}$; form $\mathbf{x}^\sigma$ and $\mathbf{v}^\star$ on generated entries ▷ Eq. (1)

$\widehat{\mathbf{v}} \leftarrow v_\theta(\mathbf{x}^\sigma, \sigma; c)$ ▷ one forward pass

$\widehat{\mathbf{x}}_{v,D} \leftarrow \mathbf{x}^\sigma_{v,D} - \sigma_b \widehat{\mathbf{v}}_{v,D}$, $\widehat{\mathbf{a}}_t \leftarrow \mathbf{a}^\sigma_t - \sigma_b \widehat{\mathbf{v}}_{A,t}$ ▷ Eq. (2)

$\widehat{d} \leftarrow \exp(\mathrm{clip}(\Pi_\phi(\widehat{\mathbf{x}}_{v,D}), \ell_{\min}, \ell_{\max})) - 1$; $\widehat{\mathbf{T}}_t \leftarrow \boldsymbol{\Delta}(\widehat{\mathbf{a}}_1) \cdots \boldsymbol{\Delta}(\widehat{\mathbf{a}}_t)$; $\widehat{\mathbf{p}}_t \leftarrow \pi_{xz}(\mathrm{trans}(\widehat{\mathbf{T}}_t))$

$\mathcal{L}_{\mathrm{FM},V}, \mathcal{L}_{\mathrm{FM},A}$ by Eq. (5); $\overline{\mathcal{L}}_{\mathrm{cpp}}$ by Eq. (14) with $\mathcal{G}_{\gamma=1}(\widehat{d})$; $\overline{\mathcal{L}}_{\mathrm{obs}}, \overline{\mathcal{L}}_{\mathrm{drv}}$ by Sec. A.3

$\mathbf{g}_A, \mathbf{g}^A_k, \mathbf{g}_D, \mathbf{g}^D_{\mathrm{cpp}}$ by differentiation to the velocity outputs ▷ Eq. (23)

$\nu_A, \nu^A_k, \nu_D, \nu^D_{\mathrm{cpp}} \leftarrow$ minibatch norms of these gradients

s_k by Eq. (24), γ by Eq. (29); set γ in $\mathcal{G}_\gamma$ ▷ constants in the backward pass

$\mathcal{L}_{\mathrm{total}} \leftarrow \lambda_V \mathcal{L}_{\mathrm{FM},V} + \lambda_A \big(\mathcal{L}_{\mathrm{FM},A} + \sum_{k \in \mathcal{K}} s_k \overline{\mathcal{L}}_k\big)$ ▷ Eq. (8)

backpropagate $\mathcal{L}_{\mathrm{total}}$ through v_θ once and update θ

whose factor λ_V/λ_A appears because CPP is weighted by λ_A in $\mathcal{L}_{\mathrm{total}}$ while its depth-side reference, the depth part of the visual flow loss, is weighted by λ_V. In terms of the norms, $\gamma = \lambda_V \nu_D \nu^A_{\mathrm{cpp}} / (\lambda_A \nu_A \nu^D_{\mathrm{cpp}})$, the reciprocal of the last two factors in Eq. (26), and does not depend on η_{cpp}.

The depth backward path. Between the CPP value and the depth velocity output of a view, the gradient passes through the unprojection, the exponential and clip of the depth head (Sec. 3.3), the fixed head, and the clean estimate of Eq. (2). Writing $\ell = \Pi_\phi(\widehat{\mathbf{x}}_{v,D})$ for the head's log-depth output and $\mathbf{J}_\Pi$ for its Jacobian, the view-v block of $\mathbf{g}^D_{\mathrm{cpp}}$ is

$$\mathbf{g}^D_{\mathrm{cpp},v} = -\sigma\, \mathbf{J}_\Pi^\top \,\mathrm{diag}\big(\mathbf{1}[\ell_{\min} < \ell < \ell_{\max}]\,\big(1 + \widehat{d}\big)\big)\, \nabla_{\widehat{d}} \overline{\mathcal{L}}_{\mathrm{cpp}}, \tag{30}$$

where $\nabla_{\widehat{d}} \overline{\mathcal{L}}_{\mathrm{cpp}}$ collects the per-cell terms of Eq. (15) with the weights of Eq. (14) and is zero at invalid cells and at latent frames j with $\kappa(j) \notin \mathcal{A}$, $1 + \widehat{d}$ is the derivative of the exponential, the indicator $\mathbf{1}[\cdot]$, equal to one inside the clip range and zero outside, is the derivative of the clip, and $-\sigma$ is the derivative of the clean estimate with respect to the velocity; cells whose log-depth leaves the clip range send no gradient. The final backward pass multiplies this vector by $\lambda_A s_{\mathrm{cpp}} \gamma$, the gain entering at $\widehat{d}$ through $\mathcal{G}_\gamma$. Because backpropagation is linear in the gradient it carries, a scalar gain anywhere on the segment from the unprojected points $\mathcal{U}_v(\widehat{d}, u)$ back to $\widehat{\mathbf{x}}_{v,D}$, which only the depth branch traverses, has this effect; a scale on the loss would also reach the motion branch, and one at $\widehat{\mathbf{v}}_{v,D}$ would also scale the flow gradient added there. Since ν^D_{cpp} is measured with $\gamma = 1$, it already includes the effect of the head, the exponential, and the clip, so Eq. (28) holds exactly without any separate correction for these factors.

One training update. Algorithm 1 lists one update. The norms are obtained by differentiating each loss only as far as the velocity outputs, at a cost independent of the transformer, and are formed over the whole minibatch, so the coefficients are common to all of it. A coefficient whose denominator vanishes is set to zero, so an inactive term drops out of the update. The gain is measured at $\gamma = 1$ and set before the single backward pass through the model, which is the only pass that reaches the parameters.

A.5 Inference and Trajectory Selection

Joint denoising and decoding. At inference, all generated representations are initialized independently from Gaussian noise and denoised jointly with the UniPC sampler [68] of the pretrained framework [2]. Current RGB, motion history, and text stay fixed at every model evaluation, and the padding channels of the action stream stay zero. The VAE decoder $\mathcal{D}$ reconstructs RGB and depth at full resolution; averaging the decoded depth channels and applying Eq. (10) gives metric depth. The depth head Π_ϕ provides geometric supervision during training only and is not used at inference.

Ego trajectory and selection. The generated pose rows are converted to relative rotations through the continuous representation [77] and composed into $\widehat{\mathbf{T}}_t$. Conjugation by the fixed front-camera calibration gives $\widehat{\mathbf{T}}^{\mathrm{ego}}_t = \mathbf{G}_{\mathrm{F}} \widehat{\mathbf{T}}_t \mathbf{G}_{\mathrm{F}}^{-1}$. With $\widehat{\mathbf{R}}^{\mathrm{ego}}_t$ its rotation, the first two translation coordinates give the planar ego position and the yaw is $\mathrm{atan2}([\widehat{\mathbf{R}}^{\mathrm{ego}}_t]_{21}, [\widehat{\mathbf{R}}^{\mathrm{ego}}_t]_{11})$. For K joint samples, let $\mathbf{y}^{(k)}_t$

be the planar ego position at time t in candidate k. We select

$$k^* = \underset{k\in\{1,\dots,K\}}{\arg\min} \sum_{l=1}^{K} \frac{1}{T} \sum_{t=1}^{T} \|\mathbf{y}_t^{(k)} - \mathbf{y}_t^{(l)}\|_2. \tag{31}$$

The score uses planar positions only; yaw, generated geometry, measured targets, and simulator feedback play no part. The selected candidate is returned in full, including its yaw, without averaging across candidates or any further tie-break, and its video and depth are the outputs of the same joint sample.

B Data and Labels

PHYSWAM is trained on recorded driving logs. Every supervision signal is derived from the sensors, calibration, poses, object annotations, and maps that the logs already contain without any additional manual annotation. This appendix describes the training windows (Sec. B.1), the dense metric depth labels of the depth streams (Sec. B.2), the LiDAR targets of CPP (Sec. B.3), and the obstacle and drivable-area labels of the hinge losses (Sec. B.4). All geometric labels of a window are expressed in $\mathcal{C}_0$, the front-camera frame at the current time, using the recorded poses, so that generated depth, generated motion, and every label share one frame.

B.1 Source Data and Training Windows

Logs and split. We use the `navtrain` split of NAVSIM [12], built on OpenScene [11] and nuPlan [5]: 1,192 logs recorded at 2 Hz. Each log frame provides eight camera images, one merged LiDAR sweep, the ego pose, the driving command, and oriented 3D boxes with track identities for all annotated objects. We use the three forward-facing cameras: front, front-left, and front-right. Camera intrinsics, Brown lens distortion, and camera-to-LiDAR extrinsics are constant per vehicle and differ between vehicles; images are used as recorded, without undistortion, and every projection in the pipeline uses the vehicle's own model.

Windows. One training window is formed at every `navtrain` scene token: the current frame, the eight following frames ($T = 8$, 4 s), and the preceding frame, which supplies the motion-history row $\mathbf{a}_0$. A total of 103,281 windows are used for training. Consecutive tokens of a log are mostly one frame apart, so windows overlap heavily; no command-based filtering or resampling is applied. The front view is used at 832×468 and the side views at 416×234.

Motion and text. The pose of the front camera at each frame is the composition of the recorded ego pose with the camera extrinsics. The relative motion $\mathbf{\Delta}_t$ of Sec. 3.1 is the pose of frame t expressed in the front-camera frame of frame $t-1$, stored as translation in metres and the first two columns of the rotation matrix. The text context is a fixed template that names the city, the discrete driving command (turn left, go straight, turn right, or follow the route), and the current speed; it is replaced by the empty string in 10% of training examples to support classifier-free guidance.

B.2 Dense Metric Depth Labels

The depth streams are generated modalities and are supervised through the flow-matching loss like RGB, which requires a dense depth video for every view and frame. LiDAR alone is too sparse for this purpose, so the labels are metric depth predictions of a feed-forward reconstruction model anchored to the LiDAR sweep of each frame. The labels serve the flow-matching loss only; CPP compares generated depth with the LiDAR itself (Sec. B.3).

Reconstruction with a LiDAR prior. For each frame of a window, MapAnything [25] is run once on the three views jointly, with metric scale enabled and given the calibrated camera poses and the frame's LiDAR sweep projected into each view as a sparse metric depth prior. The predicted metric depth and validity mask are resampled to the image grid of Sec. B.1, so that the depth label, the LiDAR targets, and the calibrated rays share one pixel grid. Sky pixels are identified by the sky probability of Depth Anything 3 [35] and set to $d_{\max}$.

Hole filling. MapAnything leaves about one fifth of the non-sky pixels of a frame invalid, concentrated in the band around the horizon. Structure for these holes is taken from the monocular metric prediction of Depth Anything 3 [35] on the same frames, scaled to the LiDAR once per window. Rather than directly use this prediction, we correct it towards the label. Where both are valid, we compute the log-ratio between the label and the monocular depth; inside a hole, this ratio is interpolated smoothly from the surrounding valid pixels, as the solution of Laplace's equation [41] with those pixels as boundary values, and the fill is the monocular depth scaled by the interpolated ratio.

Holes that the monocular model also leaves unstructured are filled by interpolating the label's own log depth in the same way, and are assigned $d_{\max}$ only where the interpolated depth exceeds 40 m. On the front view, 65% of the pixels of the final label come from MapAnything, 16% from the corrected fill, 15% are sky, and 4% are smooth extensions; the two side views are similar.

Bias correction. Compared with the LiDAR, MapAnything reads about 3% far at 10–25 m and 5% near at 60–80 m, a bias that is a function of depth alone and is consistent across vehicles. We fit a piecewise-linear correction of log depth, with the range from 2 m to 80 m partitioned into ten segments, on 34.6 M label–LiDAR pixel pairs from all sixteen vehicles, keeping 30% of the driving logs aside to check the fit, and apply it to every label before encoding. After correction, the median log-ratio between label and LiDAR is within 0.023 in every depth band.

Encoding. The corrected label is clipped to $[d_{\min}, d_{\max}] = [0.5, 80]$ m, encoded by Eq. (9), and stored as an 8-bit video at the resolution of the RGB stream; the 255 levels span the log-depth range in steps of about 1.6% relative depth.

Verification. Table 5 compares the final labels with the raw LiDAR on 1,000 windows covering all sixteen vehicles, using a projection of the sweeps written independently of the label pipeline. Agreement is measured on the LiDAR-supported latent cells that CPP uses and on all pixels with a LiDAR return within 25 m; the front–side seam is checked on the columns where the views overlap.

Table 5: **Depth-label quality against LiDAR** on 1,000 training windows. AbsRel is the mean absolute relative depth error; the cell-level column uses the geometric-mean LiDAR depth of latent cells with at least three returns, the pixel-level column all LiDAR returns within 25 m. Sky error is the fraction of pixels labelled sky that carry a LiDAR return.

View	AbsRel, cells	AbsRel, pixels ≤ 25 m	Sky error
Front	0.041	0.073	0.05%
Front-left	0.051	0.113	0.22%
Front-right	0.060	0.118	0.20%

B.3 LiDAR Targets

Projection. For every frame of a window and every view, the frame's LiDAR sweep is projected into the camera through the vehicle's own model. Returns on the ego vehicle are removed, the remainder are transformed into the camera frame and restricted to optical-axis depths in $[0.3, 80]$ m, and points outside the lens field of view are discarded before distortion is applied. Each surviving return is assigned to the pixel containing its projection, and where several returns land on one pixel the nearest is kept.

Pooling. The projected returns are pooled onto the latent grid of Sec. A.1, whose 16×16-pixel cells coincide with the cells of the VAE. For each cell we store the number of returns and the mean of their log depths; the CPP target $d_t^{L,v}(u)$ of Eq. (6) is the exponential of the latter, the geometric-mean depth, and a cell is a valid correspondence when it holds at least $n_{\min} = 3$ returns. Cells on moving objects are retained: at each anchor, generated and measured geometry describe the same instant, so the comparison is valid for dynamic and static structure alike. The same pooled targets serve to fit the depth head of Sec. A.2.

Poses and rays. The recorded poses $\mathbf{T}_t^\star$ are the front-camera poses of Sec. B.1 expressed in $\mathcal{C}_0$, and equal the composition of the recorded motion rows. The calibrated rays of Sec. A.1 are obtained by inverting the vehicle's distortion model at every cell and patch centre.

B.4 Obstacle and Drivable-Area Labels

Obstacle sets. For each future frame $t = 1, \ldots, T$, the box set $\mathcal{O}_t$ of Sec. A.3 contains all annotated objects within 40 m of the ego vehicle at t, of every class, as oriented rectangles in the ground plane of $\mathcal{C}_0$. The static set $\mathcal{S}_t$ is built from the sweep recorded at t: returns on the ego vehicle and inside any annotated box, dilated by 0.3 m, are removed; the remainder are restricted to heights between 0.3 m and 3 m above the local ground, estimated from the lowest returns near the vehicle, and to planar distances between 1 m and 25 m from the ego position at t. Isolated returns are suppressed by keeping only 0.5 m ground-plane columns with at least two returns, each represented by its return nearest to the recorded waypoint, and the set is limited to the 2,048 points nearest to that waypoint, so the return that determines the recorded clearance is always among them. The recorded clearance $d_t^\star$ is evaluated from these sets at the recorded waypoint.

Drivable-area field. The drivable region $\mathcal{D}$ is the union of the road-block, intersection, and car-park polygons of the nuPlan map within 100 m of the ego vehicle at the current frame, the same

layers that the NAVSIM drivable-area metric treats as permissible. The polygons are rasterized in the ground plane of $\mathcal{C}_0$ on the 128×128 grid of Sec. A.3 with $r_S = 0.5$ m, covering $x \in [-32, 32)$ m and $z \in [-8, 56)$ m, and the signed distance field is computed by Euclidean distance transforms of the rasterized region and its complement. On 60 held-out windows the rasterized region agrees with the NAVSIM devkit's own drivable map cell for cell.

Coverage. Every training window carries all three cameras on all frames, a LiDAR sweep on the current and future frames, calibration, annotations, and a complete set of labels; windows missing any of these are excluded rather than substituted.

C Experimental Details and Additional Results

C.1 Hyperparameters and Metrics

PHYSWAM is obtained by full fine-tuning of the generation pathway of Cosmos 3 Nano on the NAVSIM `navtrain` windows of App. B, with the understanding pathway, the VAE, and the depth head fixed. Every update minimizes the objective of Eq. (8): flow matching on the three RGB streams, the three depth streams, and the ego-motion rows, with the CPP and hinge terms scaled to a fixed fraction of the action flow-gradient norm. The hinge fractions are annealed to zero during the first half of training and CPP stays active throughout, so the geometric supervision that remains at the end is the one the model carries into planning. One model, evaluated with an exponential moving average of its weights, produces every PHYSWAM result of Sec. 4.1, and Table 6 lists its settings. The rows marked "without CPP" come from the same recipe with the CPP term removed (Sec. 4.2).

Metrics. PDMS combines no collision (NC), drivable-area compliance (DAC), time-to-collision (TTC), comfort (C), and ego progress (EP); EPDMS adds driving-direction (DDC), traffic-light (TLC), and lane-keeping (LK) compliance and splits comfort into history and extended comfort (HC, EC).

Table 6: **Hyperparameters** of PHYSWAM.

Data	
Training windows	103,281 (NAVSIM `navtrain`)
Frames per window	1 current + 8 future, 2 Hz
Views and resolution	front 832×468; front-left and front-right 416×234
Model	
Initialization	Cosmos 3 Nano (15.2B parameters)
Trained parameters	generation pathway, 7.0B
Noise-level shift	$\sigma = 5u/(1 + 4u)$, u uniform; training and sampling
Optimization	
Updates $\times$ batch	30,000 $\times$ 44 windows
Optimizer	AdamW, $\beta = (0.9,\ 0.99)$, no weight decay, clip 1.0
Learning rate	peak 2×10^{-5}, 300 warm-up updates, linear decay to 0
Precision	fp32 master weights, bf16 compute
Weight averaging	EMA, power-function profile, relative width 0.1
Objective	
λ_V, λ_A	10, 20
View–modality weights $\omega_{v,m}$	front RGB 1; side RGB $1/4$; depth $1/6$ per view
η_{cpp}	0.2 throughout
η_{obs}, η_{drv}	0.2 to update 9,000, annealed smoothly to 0 by update 15,000
CPP anchors $\mathcal{A}$; Huber transition δ	$t \in \{4, 8\}$ (2 s, 4 s); 0.5 m
Hinge margins m_{obs}, m_{drv}	1.6 m, 1.5 m
Inference	
Sampler	UniPC, 30 steps, no guidance
Trajectory selection	one sample; or medoid of K=8 samples

C.2 `navhard` Stage Scores and Sub-Scores

The two stages of `navhard` separate the quality of a plan in the recorded scene from its behaviour once traffic reacts to it. Table 7 lists both for the methods that report them. In the first stage PHYSWAM lies within the range of the other methods; in the second it has the highest no-collision, drivable-area, and time-to-collision terms and the highest stage score. Each synthetic second-stage

scene is weighted by how close its start lies to the end of the first-stage plan, and the combined score multiplies the two stages per scene before averaging, so it is not a function of the two stage means.

Table 7: **`navhard` per-stage sub-scores**, as reported; the LTF and DriveVLA-W0 (Flow checkpoint) rows are the values reported in [59], which reprints the LTF score of [6]; the DVGT-2 row is as reported in [40]. Stage 1: 450 recorded scenes. Stage 2: 5,462 synthetic scenes with reactive traffic. Bold: best per column.

Stage 1

Method	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	Stage 1
LTF [10]	96.2	79.6	99.1	99.6	84.1	95.1	94.2	97.6	79.1	–
DriveVLA-W0 [30]	96.8	83.3	99.0	99.6	84.6	95.3	96.4	97.6	78.2	–
DVGT-2 [81]	97.2	91.3	98.4	99.8	84.8	95.5	95.5	97.5	71.4	–
DriveFuture [18]	96.3	87.8	98.2	99.6	83.1	96.9	94.9	97.6	76.9	–
4D-WAM [14]	92.8	89.6	**99.7**	99.3	**98.6**	91.6	86.9	97.1	**81.3**	–
EponaV2 [59]	97.3	90.7	99.4	**100**	83.3	**97.3**	**97.3**	97.6	60.9	–
GeoWAM [40]	**97.7**	**91.5**	99.1	99.8	83.8	95.8	96.0	**97.8**	79.0	–
PHYSWAM	96.9	90.9	99.0	99.6	83.8	95.6	95.8	**97.8**	67.6	77.0
PHYSWAM, medoid-of-8	96.4	91.1	99.3	**100.0**	83.6	96.4	96.0	**97.8**	72.4	**78.3**
PHYSWAM, without CPP	97.0	90.2	99.0	99.8	83.9	95.6	95.1	**97.8**	67.6	76.6

Stage 2

Method	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	Stage 2
LTF [10]	77.8	70.2	84.3	98.1	85.1	75.7	45.4	95.7	**76.0**	–
DriveVLA-W0 [30]	76.8	64.3	79.9	98.3	89.2	75.0	46.8	95.8	53.1	–
DVGT-2 [81]	77.8	73.8	81.3	98.3	91.5	73.2	48.0	83.9	45.1	–
DriveFuture [18]	82.3	78.8	**88.1**	98.4	83.6	79.6	47.6	**97.0**	75.9	–
4D-WAM [14]	82.6	71.8	86.9	98.2	**97.6**	78.8	49.0	95.8	71.6	–
EponaV2 [59]	83.6	78.0	88.0	**98.9**	86.0	80.3	50.1	96.1	52.0	–
GeoWAM [40]	80.4	76.3	87.3	98.7	88.9	76.2	49.9	94.0	56.0	–
PHYSWAM	83.9	80.2	87.3	98.7	88.7	81.6	48.9	95.3	54.4	48.8
PHYSWAM, medoid-of-8	**85.0**	**81.7**	**88.1**	98.7	88.0	**82.2**	**50.5**	95.5	58.4	**50.3**
PHYSWAM, without CPP	83.7	79.6	86.1	98.7	89.1	80.7	48.5	95.0	53.5	47.0

C.3 HUGSIM by Difficulty, Metric, and Source Dataset

The simulator renders each source dataset's own camera rig; the model receives the render's pinhole rays, the simulator's navigation command and a 0.5 s odometry history, and returns an eight-waypoint plan every 0.5 s, which the benchmark's controller tracks. The rendered rig differs from the training rig in lens model, mounting height and appearance, and no adapter is used.

Table 8 gives every HUGSIM metric by difficulty for PHYSWAM and for the baselines whose per-metric values the benchmark reports. PHYSWAM has the highest drivable-area compliance at every difficulty and, on easy scenarios, leads every metric except comfort; on medium, hard, and extreme scenarios UniAD keeps higher no-collision and time-to-collision terms, which is where PHYSWAM's score is lost. Table 9 splits PHYSWAM by the source dataset of the reconstructed scenes: route completion and HD-Score are highest on nuScenes and Waymo and lowest on the hard and extreme KITTI-360 scenarios. HD-Score multiplies route completion by the per-step product of the no-collision and drivable-area terms with a weighted average of the time-to-collision and comfort terms, as PDMS does.

C.4 Ablation Details

LoRA setting. Table 10 lists the setting of the ablation rows in Table 3b and d: rank-128 adapters ($\alpha = 256$) on the attention projections of the generation pathway, trained for 16,000 updates. Settings not listed are as in Table 6. The reference row is flow + hinge + CPP with all three views, and every other row changes only the objective terms or inputs given in its label. Removing generated depth also removes CPP, which acts on it. Table 11 gives all sub-scores and Table 12 the scores by driving command. Without generated depth, and hence without CPP, the multi-view model scores 87.5 PDMS, 0.3 above flow + hinge with generated depth and 0.7 below the full objective. Generated depth therefore helps planning through CPP rather than on its own.

Model trained without CPP. At 4,000 updates (Table 3a, top), the plans of the model without CPP drift laterally through a less accurate heading rather than taking wrong turns. The final model without CPP also generates worse depth (Table 13). Its AbsRel against LiDAR is higher on every

Table 8: **HUGSIM by difficulty and metric** ($\times 100$); baselines as reported by the benchmark paper. The Overall rows weight the four difficulty levels by their number of episodes. Bold: best per metric and difficulty among UniAD, VAD, LTF, and PHYSWAM.

Method	Difficulty	NC	DAC	TTC	COM	RC	HD-Score
UniAD [21]	Easy	77.4	88.5	70.8	82.8	58.6	48.7
	Medium	**72.4**	86.8	**60.4**	72.0	41.2	29.5
	Hard	**66.5**	86.0	**55.0**	67.2	**40.4**	**27.3**
	Extreme	**54.4**	89.6	**42.9**	58.5	26.0	**14.3**
VAD [24]	Easy	66.1	73.9	58.2	**100.0**	38.7	24.3
	Medium	45.7	79.8	29.0	**100.0**	27.0	9.9
	Hard	44.3	81.0	28.8	**100.0**	25.5	10.4
	Extreme	36.1	83.7	25.9	**100.0**	23.0	8.2
LTF [10]	Easy	74.1	81.7	70.6	99.6	68.4	52.8
	Medium	47.4	83.2	43.3	99.7	40.7	24.6
	Hard	40.7	83.9	37.0	99.9	36.9	19.8
	Extreme	28.3	85.1	21.8	99.8	25.5	8.1
PHYSWAM	Easy	**93.1**	**99.1**	**91.7**	99.4	**93.5**	**86.9**
	Medium	51.8	**97.5**	39.9	95.6	**47.4**	**30.1**
	Hard	47.7	**96.0**	39.1	95.8	38.6	25.2
	Extreme	34.3	**93.8**	27.8	95.5	**26.1**	13.4
	Overall	54.4	96.6	46.4	96.3	48.9	35.5
PHYSWAM, without CPP	Easy	92.7	98.8	91.0	99.3	92.6	85.4
	Medium	49.3	97.4	37.3	95.6	44.9	26.8
	Hard	47.0	96.0	37.8	95.2	37.8	23.9
	Extreme	32.6	93.9	25.7	95.6	25.4	12.0
	Overall	52.8	96.5	44.5	96.2	47.5	33.4

Table 9: **HUGSIM by source dataset**, PHYSWAM: RC / HD-Score per difficulty level and the episode mean per dataset.

Dataset	Episodes	Easy	Medium	Hard	Extreme	All
nuScenes	88	94.5 / 88.4	43.7 / 27.1	59.5 / 43.1	40.6 / 27.3	56.7 / 42.9
Waymo	108	94.6 / 91.6	51.5 / 34.1	57.3 / 44.8	23.9 / 8.7	55.8 / 42.8
KITTI-360	113	87.4 / 76.5	51.6 / 39.3	14.9 / 6.1	8.0 / 0.7	41.1 / 31.0
PandaSet	127	97.4 / 91.1	42.8 / 20.1	22.5 / 6.6	32.0 / 16.9	41.3 / 24.3

Table 10: **Hyperparameters of the LoRA setting.** Adapters act on the generation pathway; rows not listed are as in Table 6.

Adaptation	LoRA, rank 128, $\alpha = 256$, on the attention projections
Trained parameters	122.7M
Updates $\times$ batch	16,000 $\times$ 44 windows
Optimizer	AdamW, $\beta = (0.9,\ 0.99)$, weight decay 0.05, clip 1.0
Learning rate	peak 2×10^{-4}, 300 warm-up updates, linear decay to 0
Weight averaging	none

Table 11: **LoRA setting, all sub-scores** on `navtest`, one sample per scene.

Row	NC	DAC	DDC	TLC	EP	TTC	LK	HC	PDMS	EPDMS
flow	95.9	96.4	98.6	99.7	86.0	94.9	95.2	98.4	85.7	84.3
flow + hinge	96.8	96.6	98.5	99.8	86.8	95.9	95.3	98.6	87.2	85.7
flow + CPP	96.8	97.0	98.5	99.7	87.0	95.7	95.1	98.6	87.8	86.2
flow + hinge + CPP (multi-view)	96.9	97.2	98.5	99.7	87.2	95.8	95.1	98.6	**88.2**	**86.5**
single-view	96.6	96.7	98.6	99.7	86.8	95.6	95.2	98.5	87.1	85.6
multi-view, no depth	96.9	96.8	98.5	99.7	87.0	95.9	95.2	98.6	87.5	85.9

view and horizon, by 8–15% at the pixel level and 5–7% on the cells that CPP supervises, and its $\delta_{1.25}$ is lower on every view and horizon except the front view at +4 s, where the two are equal within sample noise. In the front view at +2 s, AbsRel is 0.193 against 0.175. Of its sampled futures, 70.9% end at +4 s within 10° of the recorded heading and within 0.3 m of the recorded position along the forward axis, against 81.4% with CPP. Video quality (FVD, FID) is unchanged within sample noise. Removing CPP from update 20,000 onward leaves open-loop scores within sample noise but

Table 12: **Scores by driving command** on `navtest` (2,501 left, 8,070 straight, 1,575 right scenes), one sample per scene.

	PDMS			EPDMS		
Row	Left	Straight	Right	Left	Straight	Right
flow	81.4	87.8	81.9	80.3	86.2	81.3
flow + hinge	83.4	89.0	83.8	82.4	87.2	82.8
flow + CPP	83.5	89.9	83.9	82.1	88.1	83.0
flow + hinge + CPP (multi-view)	83.9	90.3	84.3	82.4	88.4	83.3
single-view	81.2	90.0	82.0	80.1	88.2	81.4
multi-view, no depth	83.5	89.4	84.0	82.4	87.6	83.0
PHYSWAM (full fine-tune)	87.8	92.7	90.4	87.3	91.4	89.4

Table 13: **Generated depth with and without CPP** for the final models, on `navtest`: AbsRel and $\delta_{1.25}$ against the LiDAR of the same future frame and camera, without scale alignment, as in Table 3a; means over four samples per scene. "Cells": the 16-pixel cells that CPP supervises.

	+2 s				+4 s			
	AbsRel↓		$\delta_{1.25}$ ↑		AbsRel↓		$\delta_{1.25}$ ↑	
View	with	without	with	without	with	without	with	without
Front	0.175	0.193	0.814	0.802	0.232	0.253	0.742	0.743
Front-left	0.292	0.315	0.725	0.706	0.374	0.404	0.654	0.649
Front-right	0.240	0.277	0.752	0.725	0.305	0.346	0.676	0.663
Front, cells	0.144	0.154	0.856	0.844	0.194	0.205	0.795	0.795

costs 1.2 HD-Score and 1.1 RC on HUGSIM, and raises the share of wrong-turn futures, those with heading error above 40° at +4 s, from 0.4% to 0.6%.

Hinge schedule. Both hinges are one-sided and vanish on every plan that keeps its margins, and the obstacle margin is capped at the recorded clearance (Sec. 3.4), so the recorded trajectory incurs no obstacle loss. Early in training many plans violate the margins, and the hinges give a direct gradient on waypoint positions. Late in training few plans do, yet the balancing of Sec. 3.4 still scales each active term to a fixed fraction of the action flow-gradient norm, which would concentrate that fraction on the few remaining plans. We therefore anneal both hinges to zero by the middle of training. CPP penalizes deviation in either direction on every LiDAR-supported cell, so it does not vanish once plans are safe.

C.5 World-Model Quality and Consistency

Table 14: **Front-view video quality on NAVSIM.** I3D FVD over one context and eight generated frames at 2 Hz, FID over the same frames. 600 clips: generated clips of half the scenes against the recorded clips of the other half, ten splits, floor from the two recorded halves; 1,200 clips: generated against recorded clips of the same scenes. Reported rows use their own clip counts and lengths (DrivingGPT [8] 12 frames; PWM [69] 10, † as reported by DriveDreamer-Policy [76], itself 9; CoWorld-VLA [22] unstated).

	Clips	FVD↓	FID↓
DrivingGPT	512	142.6	12.8
PWM†	–	86.0	–
DriveDreamer-Policy	–	53.6	–
CoWorld-VLA	–	32.7	–
Recorded (floor)	600	$91.5_{\pm 3.2}$	$11.0_{\pm 0.3}$
PHYSWAM	600	$111.3_{\pm 6.8}$	$15.4_{\pm 0.6}$
PHYSWAM	1,200	42.2	6.8

Metric depth. Decoded from a sample and scored against the LiDAR sweep of the same future frame and camera without any scale alignment, front-view depth reaches AbsRel 0.175 at +2 s and 0.232 at +4 s, with $\delta_{1.25}$ of 0.814 and 0.742 (Table 3c); on the cells that CPP supervises, AbsRel is 0.144 and 0.194. The side views, generated at half resolution, have 1.3–1.7× the front error (Table 13), and the median signed error is negative in every front-view setting, placing generated surfaces 0.2–0.3 m nearer than the LiDAR. Because each sample renders the model's own future viewpoint, these errors include any difference between the generated and the recorded ego motion. Reported future-depth results on nuScenes (Table 3c, lower block) place GeoWAM [40] at AbsRel 0.245 and 0.297 at the two horizons, Epona [66] with a DVGT [80] geometry head at 0.263 and

0.310, VGGT-World [47] at 0.329 and 0.357, and the same base model as ours with that head, Cosmos 3 [2] + DVGT, at 0.376 and 0.422; the front-view values of PHYSWAM are lower at both horizons, on a different dataset and rig.

Video. Table 14 gives the front-view video scores behind Sec. 4.3. At 600 clips per side, the generated front video reaches FVD-9 111.3 against a recorded-versus-recorded floor of 91.5, an excess of 19.8 that is positive on all ten splits, and FID 15.4 against 11.0; over all 1,200 clips it reaches 42.2 and 6.8 (Table 14). Reported NAVSIM front-view results range from FVD 142.6 for DrivingGPT [8] on 512 clips to 32.7 for CoWorld-VLA [22], each under its own clip count and length; on nuPlan, the source data of NAVSIM, Epona [66] reports 50.8 over 1,628 ten-frame clips. At a similar clip count, PHYSWAM is below DrivingGPT, and the recorded floor shows how much of any reported value is sample size: the gap between generated and recorded clips is about a fifth of the floor itself.

Table 15: **Consistency of the generated future** with its own motion, across views, and over time, on `navtest`; medians. Yaw: camera rotation recovered from the generated video [25] against the generated motion, and, for the floor, from the recorded clip against the recorded trajectory; the excess is the median of the per-scene paired differences, with a 95% bootstrap interval. Depth: median $|\Delta \log z|$ after warping the front depth into the side views with the recorded extrinsics, or frame 0 into frame 8 with the generated motion; floors from two LiDAR sweeps 0.5 s apart and from recorded LiDAR under recorded motion.

Quantity	Floor	PHYSWAM	Excess
Yaw, video vs. motion, 4 s (°)	0.29	0.80	+0.44 [+0.39, +0.48]
turns above $45°$	6.11	–	+2.31 [+1.93, +2.86]
Cross-view $\|\Delta \log z\|$, +2 s, L / R	0.032 / 0.030	0.045 / 0.049	1.5×
Temporal $\|\Delta \log z\|$, 0 to 4 s	0.087	0.285	3.3×

Consistency with the generated motion. Table 15 gives the floors and excesses behind the consistency line of Sec. 4.3. The camera yaw that MapAnything [25] recovers from each generated video deviates from the generated motion by a median of $0.80°$ over 4 s, against $0.29°$ when it is run on recorded clips, and where the generated motion departs from the recorded trajectory the video follows the generated motion. The generated front depth agrees with the generated side depth to within 1.5× the disagreement between two LiDAR sweeps 0.5 s apart, and frame to frame under the generated motion the disagreement grows to 3.3× the LiDAR floor over 4 s. The yaw excess over the 4 s horizon is $0.44°$ with a 95% bootstrap interval of $[0.39, 0.48]$; above $45°$, where the median commanded turn is $58.7°$, it is $2.31°$. The excess concentrates in sharp turns, where the video realizes about 96% of a hard turn; where the generated motion and the recorded trajectory differ by more than $2°$, the video follows the generated motion in 67% of the scenes. Warped into the side views with the recorded extrinsics, the generated front depth disagrees with the generated side depth by a median $|\Delta \log z|$ of 0.045–0.049 at +2 s, about 1.5× the disagreement between two LiDAR sweeps 0.5 s apart. Warped from frame to frame with the generated motion, the per-step depth disagreement stays between 0.038 and 0.043 and accumulates close to linearly to 0.285 over 4 s, 2.3–3.3× the LiDAR floor along the horizon. At every step the generated depth agrees more closely with the generated motion than with the recorded motion, by about 20%.

C.6 Sampler Steps and Cost

Table 16 varies the sampler steps at inference. The step count moves PDMS by less than a point and not monotonically: 8 steps score 0.7 above the 30-step protocol and the single 4- and 15-step samples 0.6 and 0.4 below, against a sample-to-sample sd of 0.2–0.3, while the cost of a plan grows from 3.5 to 9.4 GPU-seconds per scene. All other results in this paper use 30 steps.

C.7 Additional Qualitative Examples

Figure 4 compares the model with CPP against the model trained without CPP, and the multi-view model against the single-view model, on two further scenes each. It also shows two scenes in which the full model fails.

D Limitations

Supervision. CPP compares generated geometry with LiDAR sweeps carried by the recorded poses, so training needs a LiDAR sensor calibrated to the cameras and an accurate ego trajectory. The depth streams need dense metric labels, which we obtain from a reconstruction model anchored to the same LiDAR (App. B.2), and the hinges need obstacle and drivable-area labels (App. B.4). Every term of the objective is therefore tied to an instrumented collection platform. Scaling to the far larger body

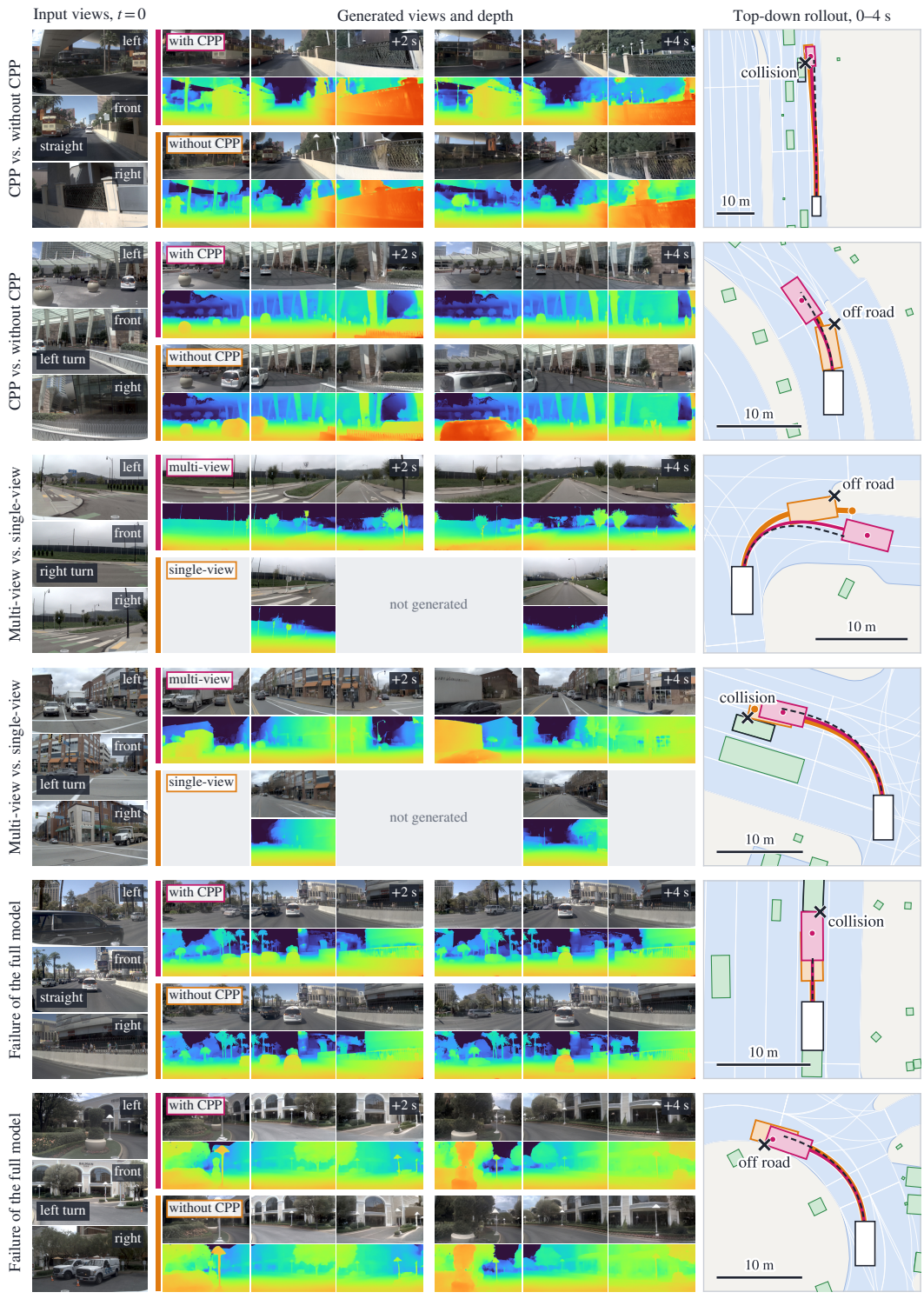

Figure 4: **Additional qualitative examples.** On a straight road, the model with CPP generates the overtaking bus alongside in its left view. The model trained without CPP omits it and is clipped by the overtaking car at 3.9 s. At a hotel drop-off, it generates the departing minivan too close and clips the island at 1.6 s. The single-view model swings wide over a grass median that only the right camera shows and leaves the road at 3.1 s. At a left turn, it never generates the waiting truck and hits it at 3.7 s. The full model renders a stopped minivan at a constant size, plans as if it were pulling away, and rear-ends it at 3.7 s, a world-model error. At a tight left turn, both models generate indistinguishable views, but the full model clips the inner curb by 0.4 m at 3.6 s, a plan-precision error.

Table 16: **Sampler steps and cost** for the model of Tables 1–2: PDMS relative to the 30-step protocol row, one sample per scene, averaged over the number of samples given (sample-to-sample sd 0.2–0.3 where repeated), and the cost of one plan on one RTX PRO 6000 GPU.

UniPC steps	ΔPDMS	Samples	GPU-s / scene
4	−0.6	1	3.5
8	+0.7	3	4.4
15	−0.4	1	6.0
30	0	4	9.4

of driving video recorded without LiDAR would need a self-supervised form of the coupling, for instance between generated depth and generated motion alone, which we have not developed.

Consistency against the recorded future only. CPP enforces agreement between generated depth and motion where LiDAR supports the recorded trajectory. Self-consistency away from that trajectory, under counterfactual commands or beyond the 4 s window, is measured only indirectly (App. C.5) and is not supervised, and whether the coupling generalizes there is open.

One backbone, one scale, one training run per configuration. All results come from Cosmos 3 Nano at one model size, and each training configuration was run once. We report sample-to-sample variation (Table 1) but not run-to-run variation, and we have not studied how the formulation behaves across backbones, model sizes or data scales due to compute constraints.

Inference cost. Every plan generates a full multi-view future. One plan costs 9.4 GPU-seconds at 30 sampler steps and 4.4 at 8 (App. C.6), the medoid rule multiplies this by the number of samples, and an action-only inference path was not tested.

E Extended Related Work

Table 17: **World modeling and its role in planning.** Representative works illustrate how future prediction supports policy learning, action generation, or trajectory selection. These roles can overlap within a method.

Representative works	Predicted world	Connection to action
LAW, SimWAM [28; 70]	Future features or video	Future prediction provides training supervision; planning does not require explicit future generation.
DriveWAM, GeoWAM [45; 40]	Future video or 3D geometry	Representations of the predicted future condition trajectory generation.
Epona, DriveVA [66; 38]	Future video	Video and action generation share learned representations.
4D-WAM [14]	Future video	Joint video–action learning is supervised by matching geometry recovered from generated and recorded video.
WoTE, DA-WAM [29; 73]	Future scene features	A learned scorer evaluates candidate trajectories using their predicted future states.
PHYSWAM (ours)	Multiview video and metric depth	Video, depth, and ego motion are co-denoised. CPP jointly supervises generated depth and motion against measured scene geometry. A label-free consensus rule selects among trajectories.

Table 17 summarizes how representative works connect future prediction to planning; the paragraphs below discuss them in turn.

Generative driving world models. Driving world models learn controllable future-video generation from driving data [55; 19; 16], with GAIA-2 supporting coordinated generation across multiple cameras [43]. Other representations make scene structure explicit: MUVO predicts images, LiDAR, and occupancy [4], while OccWorld jointly forecasts occupancy and ego motion [71]. OmniNWM further combines panoramic video with depth, semantics, and occupancy [26]. These approaches establish the value of modeling the future beyond appearance (RGB video) alone. PHYSWAM combines multiview video and metric depth with ego-motion generation in a shared flow-matching model, providing explicit scene geometry alongside the generated action.

World–action models (WAMs) and planning. Future prediction can improve policy representations through auxiliary visual or latent objectives [28; 70]. Joint video–action modeling has also been explored in driving [66; 38] and robotics [79; 63]. Beyond generation, predicted visual or latent futures support learned trajectory evaluation and policy refinement [72; 60; 37]. ExploreVLA uses uncertainty in future predictions to guide safety-aware policy exploration [44]. PHYSWAM follows the joint-generation approach while keeping trajectory selection simple: a parameter-free

consensus rule uses only pairwise distances among sampled trajectories, without a learned scorer or simulator feedback.

Geometry in world and action generation. View synthesis provides a foundation for jointly learning depth and camera motion through geometric supervision [75]. In generative models, geometry serves as a conditioning signal, an output, or a training target. GeoDrive and ReCamDriving use geometry rendered along supplied trajectories to guide video generation [7; 62]. UniFuture jointly forecasts RGB and depth [32], while X-WAM adds depth prediction to video–action modeling [17]. DriveDreamer-Policy integrates depth, video, and action experts [76], and GeoWorldAD and GeoWAM use geometric representations to condition planning [67; 40]. Geometry Forcing aligns video-model features with geometric representations [57], while 4D-WAM supervises generated video through depth and features recovered from generated and recorded frames [14] via a Geometric Foundation Model [49; 50]. PHYSWAM explicitly includes generated ego motion in its geometric objective. CPP unprojects generated depth, transforms the resulting geometry using generated motion, and compares it with LiDAR transformed using recorded motion in a shared reference frame. This shared metric loss promotes physical consistency with the measured scene by supervising both outputs alongside their individual flow-matching objectives.