STREAMDRIVE: STREAMING PERCEPTION FOR END-TO-END AUTONOMOUS DRIVING VIA DELAY-CONDITIONED FEATURE EXTRAPOLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

End-to-end autonomous driving systems achieve strong offline benchmarks but ignore a real-world constraint: by the time inference completes, the world has moved on and predictions are stale. We present StreamDrive, a streaming perception framework that equips end-to-end driving models with delay-aware feature extrapolation. StreamDrive corrects *latent instance features* rather than individual task outputs: a single delay-conditioned residual propagates through the entire task stack: detection, motion forecasting, and planning. A timestamp-aware temporal queue replaces frame-indexed storage with wall-clock timestamps, keeping instances matched under variable frame rates and irregular sensor timing. StreamDrive decomposes delay compensation into a constant-velocity baseline that captures first-order motion, plus a FiLM residual that learns non-linear, delay-dependent corrections conditioned on scene context and ego kinematics. On nuScenes, StreamDrive improves streaming detection mAP by $+1.3$ pp at real-time ($1\times$) playback over the SparseDrive-S baseline. Streaming planning collision rate drops by 9.4% relative at $1\times$ and 14.2% at $4\times$ playback speed, with consistent gains across all tasks and playback speeds. We also introduce a multi-task streaming evaluation framework for end-to-end driving, including streaming metrics for motion forecasting and the Plan Consistency Score (PCS), a time-adjacent measure of plan stability under latency that complements accuracy metrics. StreamDrive adds less than 1% parameter overhead and less than 1 ms of inference latency.

1 INTRODUCTION

End-to-end (E2E) autonomous driving models, ranging from UniAD (Hu et al., 2023) and VAD (Jiang et al., 2023) to SparseDrive (Sun et al., 2025), have made rapid progress by jointly optimizing detection, mapping, motion forecasting, and planning in a single architecture. These systems are evaluated on standard benchmarks (Caesar et al., 2020) under an implicit assumption: predictions are available *immediately* after the input is captured. In practice, by the time a model finishes processing a frame at t, the vehicle and surrounding agents have already moved, and the prediction refers to a past world state. At a typical urban speed of 80 km/h, a 250 ms inference delay corresponds to 5.6 m of vehicle travel, which is more than a car length of unaccounted motion.

Li et al. (2020) introduced streaming perception to measure this effect for 2D detection: a model is evaluated at time $t + \mathrm{dt}$ where dt is its inference latency. Under this metric, SparseDrive-S loses 88.8% of its detection mAP (from 0.418 to 0.047) under $4\times$ playback, i.e., an effective delay of 0.5 s (Fig. 1). Recent work has begun to address streaming 3D detection (Peng et al., 2025; Li et al., 2024), but these methods are limited to a single task and do not consider the full E2E driving stack.

We present **StreamDrive**, a streaming perception framework that addresses delay-induced staleness through delay-conditioned feature extrapolation on *latent instance features*. Unlike prior streaming methods that apply per-output corrections, StreamDrive intercepts and corrects features before they reach any task head, so one correction covers the entire driving stack. It introduces three lightweight components: a timestamp-aware temporal queue, a FiLM(dt)-conditioned residual with sinusoidal delay encoding, and a scene aggregator with ego state encoder. The FiLM residual is zero-initialized (Zhang et al., 2023), so it is the identity at $\mathrm{dt}{=}0$ and offline performance is preserved

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

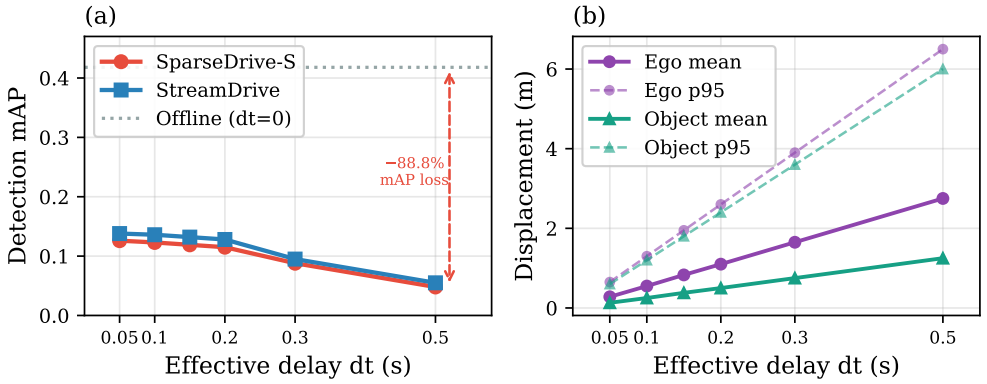


Figure 1: **Streaming degradation.** (a) Detection mAP of SparseDrive-S and StreamDrive under streaming evaluation vs. effective delay dt (dense 0.05–0.5 s sweep), with offline (dt=0) baseline shown as reference. At dt=0.5 s (4× playback), SparseDrive-S mAP drops by 88.8%. (b) Ego and object displacement over the same delays.

exactly. Training targets are computed in the future ego frame, so one residual corrects agent dynamics and ego-motion together. The design attaches to any E2E model exposing instance-level features; we demonstrate portability on a second backbone (UniAD) in Sec. 5.

Beyond the method, we contribute a **multi-task streaming evaluation framework** for E2E driving. Prior streaming work evaluates only detection; we introduce streaming minFDE/minADE for motion, streaming L2 error and collision rate for planning, and the Plan Consistency Score (PCS), a time-adjacent plan stability metric. Our contributions are: (1) a delay-conditioned feature extrapolation framework with zero offline degradation (Sec. 3); (2) a timestamp-aware temporal queue that keeps instances matched under variable latency (Sec. 3.3); and (3) this multi-task streaming evaluation framework (Sec. 4). Absolute numbers in the streaming regime are low for *all* methods, so we report paired bootstrap intervals for all gains (Sec. 5.4).

2 Related Work

Streaming Perception. Li et al. (2020) introduced streaming accuracy, evaluating detectors at output time rather than input time. Ge et al. (2021) showed faster inference partially mitigates staleness but without modeling delay–quality trade-offs. ASAP (Wang et al., 2023b) extended streaming evaluation to 3D detection on nuScenes, though treating delay as fixed rather than a continuous conditioning variable. Peng et al. (2025) proposed LASP for latency-aware streaming 3D detection with temporal feature alignment. Zhang et al. (2025) addressed streaming via prediction diversity at the output level. All prior work corrects detection outputs *only*, repeating the correction per downstream task; StreamDrive is the first to extend streaming correction to the full E2E driving stack, where a single feature-level correction propagates through the shared latent representation.

End-to-End Autonomous Driving. UniAD (Hu et al., 2023) unified detection, mapping, motion forecasting, and planning under a planning-oriented framework, and SparseDrive (Sun et al., 2025) introduced sparse representation with state-of-the-art efficiency. BEVFormer (Li et al., 2022) established BEV representation via spatiotemporal transformers, though its fixed-frame assumptions break under variable-latency streaming. Sparse4D (Lin et al., 2022) introduced sparse spatial-temporal fusion, a direct precursor to SparseDrive's instance representation. The recent follow-up VADv2 (Chen et al., 2024) extends this line of work, yet all these systems evaluate under the zero-latency assumption. AD-MLP (Zhai et al., 2023) showed ego kinematics alone provide a strong planning baseline, motivating our ego state encoder. StreamDrive addresses this shared blind spot with lightweight delay compensation that leaves the frozen backbone's offline performance intact.

Delay Compensation. Perez et al. (2018) introduced FiLM for conditioning neural features via learned scale/shift. Vaswani et al. (2017) introduced sinusoidal positional encoding, later adapted by Ho et al. (2020) for continuous timestep conditioning; we apply the same principle to delay

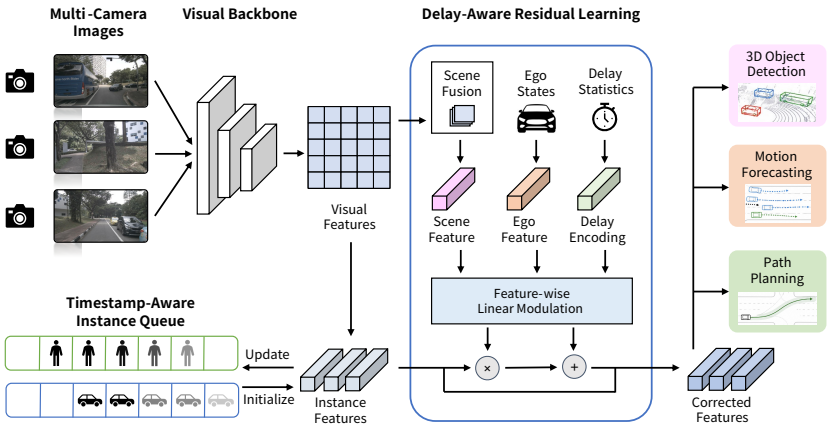


Figure 2: **StreamDrive architecture.** Lightweight residual modules are added on top of the frozen SparseDrive backbone. The FiLM(dt) layer generates scale γ and shift β from sinusoidal delay encoding, modulating instance features. Scene aggregator and ego state encoder provide additional conditioning. Zero-initialization ensures identity at $\mathrm{dt}{=}0$.

encoding. Zhang et al. (2023) demonstrated zero-initialized residuals for additive extensions to pretrained models, directly analogous to our setting where zero-initialization preserves a frozen driving backbone. StreamDrive unifies these three mechanisms: sinusoidal delay encoding gives a continuous, multi-scale representation of latency; FiLM maps it to instance-level scale/shift; and zero-initialization makes the module an identity map at zero delay. This unification is lightweight ($<$1% overhead) and applicable to any architecture that exposes instance features.

3 Method

3.1 Problem Formulation

Let $\mathbf{x}_t$ denote multi-camera images at time t and $\mathbf{y}_t$ the ground-truth labels. An E2E model f_θ produces $\hat{\mathbf{y}}_t = f_\theta(\mathbf{x}_t)$, but under streaming evaluation this prediction is not available until $t + \mathrm{dt}$ where $\mathrm{dt} > 0$ is the *inference latency*—the wall-clock time between frame capture and prediction availability (156.8 ms measured for SparseDrive-S on our hardware, varying per frame). At that point the ground truth has shifted to $\mathbf{y}_{t+\mathrm{dt}}$, so the effective error is measured against the current world state. Streaming benchmarks parameterize difficulty by *playback speed* s: the log is replayed s times faster than real time, so the recorded world advances by $s\,\mathrm{dt}$ during inference. The total world-time staleness combines two terms: t_{small}, the interval between keyframe capture and the moment the model receives the frame, and the latency dt scaled by playback speed. We write this as $\Delta = t_{\mathrm{small}} + s\,\mathrm{dt}$ and evaluate at offline ($\mathrm{dt}{=}0$), $1\times$, $2\times$, and $4\times$, where $4\times$ is hardest. The goal is $\hat{\mathbf{y}}_{t+\Delta} \approx \mathbf{y}_{t+\Delta}$, meaning accurate predictions at the time they become available. Unlike standard offline evaluation, the model must anticipate how the world evolves during its own computation; the prediction horizon is the effective staleness Δ itself.

3.2 StreamDrive Framework

StreamDrive is a streaming perception framework built on SparseDrive (Sun et al., 2025), which uses a sparse scene representation to unify detection, mapping, motion forecasting, and planning. The SparseDrive backbone consists of a multi-camera image backbone ResNet-50 (He et al., 2016) that extracts visual features, an instance queue that initializes and refines object queries via sparse interaction with these visual features, and task-specific heads. Delay compensation operates on *latent instance features* rather than individual task outputs, so a single correction reaches all downstream tasks. Correcting at the feature level also keeps the tasks consistent with each other. Detection, motion, and planning are decoded from the same corrected instance, so they cannot disagree about where an agent will be; output-level correction offers no such guarantee, since a shifted box and an unshifted trajectory can place the same agent at two different positions. Because the heads are frozen

and share this representation, the residual is applied once per instance per frame rather than once per task. StreamDrive adds three lightweight modules *on top of the frozen backbone*: a timestamp-aware temporal queue, a FiLM(dt)-conditioned residual, and a scene aggregator with ego state encoder (Fig. 2). All three modules operate on the instance features $\mathbf{f}_i$ extracted by the backbone. Concretely, compensation is applied *after* the backbone which includes its deformable image cross-attention and iterative query refinement running at the original sensor time t, and *before* the frozen task heads. Delay-shifted queries are never fed back into image cross-attention, so there is no mismatch between queries and image evidence captured at time t. The total parameter overhead is under 1% of the backbone, and the additional inference latency is $<$1 ms.

3.3 Timestamp-Aware Temporal Queue

SparseDrive (Sun et al., 2025) maintains a fixed-length instance queue indexed by frame number, assuming constant frame rate. Under streaming evaluation this breaks down: latency varies across frames, and frame-indexed queues cannot handle frame drops or multi-sensor asynchrony.

We replace frame indices with wall-clock timestamps. Each queue entry stores $(\mathbf{f}_i, t_i)$, with per-instance temporal offsets:

$$\Delta t_i = (t + \mathrm{dt}) - t_i \tag{1}$$

where t is the current frame's capture time and dt is the inference delay.

Instance correspondence is maintained via bipartite matching using spatial proximity and feature similarity, with unmatched detections creating new entries and unmatched queue entries receiving a decayed confidence. When an instance has no observation for a given timestamp, its feature persists with increasing Δt_i, so the FiLM residual extrapolates from stale observations. The per-instance delays Δt_i feed directly into FiLM conditioning (Sec. 3.4). On the first frame, all $\Delta t_i = \mathrm{dt}$ and the residual reduces to single-frame inference. Wall-clock timestamps also decouple the queue from any specific sensor rate and support asynchronous multi-sensor setups.

3.4 Delay-Conditioned Feature Extrapolation

Constant-Velocity Baseline. Given instance position $\mathbf{p}_i(t)$ and estimated velocity $\mathbf{v}_i$, the baseline extrapolates forward by dt:

$$\hat{\mathbf{p}}_i(t{+}\mathrm{dt}) = \mathbf{p}_i(t) + \mathbf{v}_i \cdot \mathrm{dt} \tag{2}$$

where $\mathbf{v}_i$ is estimated as the finite difference between the two most recent positions in the instance queue: $\mathbf{v}_i = (\mathbf{p}_i(t) - \mathbf{p}_i(t - \Delta t_{\text{prev}}))/\Delta t_{\text{prev}}$, with Δt_{prev} being the wall-clock interval between observations. This captures dominant linear motion but cannot model higher-order dynamics (acceleration, turning, interactions). The CV prior provides a strong inductive bias (Table 5); the FiLM residual learns the correction beyond this first-order baseline. The velocity computation relies on wall-clock intervals rather than assumed frame rates, so the estimate stays consistent under variable latency.

FiLM(dt) Residual with Integration. StreamDrive combines a first-order motion prior with learned non-linear corrections. The CV position estimate $\hat{\mathbf{p}}_i(t{+}\mathrm{dt})$ is encoded into a displacement feature $\mathbf{z}_i^{\text{cv}} = \mathrm{MLP}_{\text{cv}}(\hat{\mathbf{p}}_i(t{+}\mathrm{dt}) - \mathbf{p}_i(t))$, which is concatenated with the original instance feature as $[\mathbf{f}_i; \mathbf{z}_i^{\text{cv}}]$. A FiLM (Perez et al., 2018) layer conditioned on dt then produces scale and shift parameters that modulate this pair using the delay, the scene context, and the ego state. Ablations confirm that removing FiLM retains the CV benefit while the full residual branch is essential (Table 4).

Following Zhang et al. (2023), the last linear layer of the FiLM MLP is initialized to zero, so at $\mathrm{dt}{=}0$ the residual reduces to an identity transformation ($\hat{\mathbf{f}}_i = \mathbf{f}_i$) and offline performance is preserved exactly. SparseDrive's anchors and reference-point geometry are *not* independently re-extrapolated: they are time-shifted only through the corrected features $\hat{\mathbf{f}}_i$ that feed the frozen heads.

Sinusoidal Delay Encoding. The scalar delay dt is mapped to a $d{=}128$-dimensional sinusoidal embedding (Vaswani et al., 2017; Ho et al., 2020): $\mathbf{e}(\mathrm{dt}) = [\sin(\omega_k \mathrm{dt}), \cos(\omega_k \mathrm{dt})]_{k=0}^{d/2-1}$ where $\omega_k = 10000^{2k/d}$. A two-layer MLP (hidden dim 256, ReLU) produces scale $\gamma(\mathrm{dt})$ and shift $\beta(\mathrm{dt})$

that modulate instance features via a FiLM-style residual:

$$\hat{\mathbf{f}}_i = \mathbf{f}_i + \gamma(\mathrm{dt}) \odot \mathbf{f}_i + \beta(\mathrm{dt}) \tag{3}$$

where $\odot$ denotes element-wise multiplication. Sinusoidal encoding is chosen over a learned scalar embedding because it provides continuous, smooth interpolation across the delay range and encodes both short and long timescales through the geometric progression of frequencies ω_k, so corrections remain well-behaved at any dt within the operating range.

Scene and Ego Conditioning. Instance features are aggregated via mean-pooling into a scene context vector $\mathbf{s} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{f}_i$, concatenated with the delay embedding before the FiLM MLP. Vehicle kinematics (speed, acceleration, yaw rate) from CAN-bus are encoded through a small MLP and injected as additional conditioning, so the residual can distinguish scenarios where the same delay demands different corrections (e.g., braking at an intersection vs. cruising on a highway). The scene context captures surrounding agent density and layout; ego kinematics capture the vehicle's own dynamic state.

3.5 Training in the Future Ego Frame

Training targets are computed at time $t + \mathrm{dt}$ rather than t, so the residual must simultaneously account for agent dynamics and ego-motion. The multi-task loss is:

$$\mathcal{L} = \lambda_{\mathrm{det}}\mathcal{L}_{\mathrm{det}} + \lambda_{\mathrm{mot}}\mathcal{L}_{\mathrm{mot}} + \lambda_{\mathrm{plan}}\mathcal{L}_{\mathrm{plan}} \tag{4}$$

where each task loss uses the same formulation as the base model (detection focal loss, motion trajectory loss, planning L2/collision loss) but with ground truth labels transformed to the $t+\mathrm{dt}$ ego frame using known ego-motion from nuScenes odometry. During training, dt is sampled uniformly from $[0, 0.5]$ s per sample, covering the full operating range.

4 Streaming Evaluation Framework

We introduce a multi-task streaming evaluation framework for end-to-end autonomous driving. Prior streaming perception work (Li et al., 2020; Wang et al., 2023b; Peng et al., 2025) evaluates detection only; motion forecasting and planning go unevaluated under realistic latency. We extend streaming evaluation to motion forecasting and planning. Scoring all tasks under one delay protocol quantifies how errors compound across the stack and whether a single feature-level correction addresses all of them.

Streaming Detection and Motion Forecasting. Following Li et al. (2020), streaming detection evaluates predictions at time $t + \mathrm{dt}$ against ground truth at $t + \mathrm{dt}$, using standard nuScenes detection metrics (mAP, NDS) computed under simulated delay. Streaming difficulty is parameterized by playback speed $s \in \{1\times, 2\times, 4\times\}$: the log is replayed s times faster than real time, so a fixed inference latency dt induces an effective world-time staleness of $\Delta = t_{\mathrm{small}} + s\,\mathrm{dt}$ (Sec. 3). In our setup the three playback conditions are scored at effective delays of approximately 0.1 s, 0.2 s, and 0.5 s; Sec. 5.7 additionally evaluates a dense delay grid $\mathrm{dt} \in \{0.05, 0.1, 0.15, 0.2, 0.3, 0.5\}$ s. For motion forecasting, standard evaluation compares predicted trajectories against ground-truth agent positions at fixed future horizons, but under streaming conditions the prediction is available at $t + \mathrm{dt}$ and was generated from observations at t. We introduce streaming minFDE and minADE:

$$\text{s-minFDE}(\mathrm{dt}) = \min_k \|\hat{\mathbf{p}}_k^{(T)} - \mathbf{p}_{\mathrm{GT}}^{(T+\mathrm{dt})}\|_2 \tag{5}$$

where k indexes the K predicted modes, T is the forecasting horizon, and the ground truth is shifted by dt to reflect the delay. Analogously, s-minADE averages over all time steps in the horizon. They capture a failure mode that offline metrics miss: under latency, a plausible trajectory may describe where an agent *was* rather than where it *is*.

Streaming Planning and Plan Consistency. Planning is evaluated with streaming L2 error (s-L2) and streaming collision rate (s-ColRate), both computed with the ego plan displaced by dt:

$$\text{s-L2}(\mathrm{dt}) = \|\hat{\tau}(t{+}\mathrm{dt}{:}t{+}\mathrm{dt}{+}T) - \tau_{\mathrm{GT}}(t{+}\mathrm{dt}{:}t{+}\mathrm{dt}{+}T)\|_2 \tag{6}$$

$$\text{s-ColRate}(\mathrm{dt}) = \mathbf{1}[\mathrm{collision}(\hat{\tau}(t{+}\mathrm{dt}), \mathcal{W}_{t+\mathrm{dt}})] \tag{7}$$

where $\mathcal{W}_{t+\mathrm{dt}}$ is the world state at time $t+\mathrm{dt}$ and $\mathbf{1}[\cdot]$ is the indicator function (1 if its argument holds, 0 otherwise); s-ColRate averages it over evaluation frames. Beyond accuracy, we introduce PCS to measure plan *stability* under latency. PCS asks whether the planner changes its mind simply because inference took longer, which produces discontinuous control even when each plan is individually accurate. We compare two plans that describe the same absolute future time, one made at t and one made a delay later at $t+\mathrm{dt}$, and score how far apart they place the ego at that shared time:

$$\mathrm{PCS}(\mathrm{dt}) = \exp\Big(-\frac{1}{T}\sum_{s=1}^{T} \big\|\hat{\tau}_{t+\mathrm{dt}}(s) - \hat{\tau}_t(s+\mathrm{dt})\big\|_2^2 \Big) \tag{8}$$

where $\hat{\tau}_{t+\mathrm{dt}}(s)$ is waypoint s of the plan produced at $t+\mathrm{dt}$ and $\hat{\tau}_t(s+\mathrm{dt})$ is the waypoint the earlier plan assigned to that same absolute time; PCS $\in (0,1]$ (1.0 = perfect consistency) decreases monotonically with dt and must be interpreted alongside accuracy.

Protocol and ground truth. The protocol is strictly output-side and *never synthesizes ground truth*. It runs in three steps.

1. **Ingest.** The model receives the latest captured keyframe t_k, which is already t_{small} old by the time it arrives.
2. **Infer.** The model produces a prediction after its measured latency dt, so the prediction is treated as actuated at $t_k + t_{\mathrm{small}} + \mathrm{dt}$.
3. **Score.** The prediction is compared against the next *real* nuScenes keyframe t_j whose timestamp matches that actuation time, that is, one whose staleness matches $\Delta = t_{\mathrm{small}} + s\,\mathrm{dt}$.

Boxes, trajectories, and collision checks all use real annotated keyframe states; no labels are interpolated at non-annotation timestamps (the dense sweep uses the keyframe-derived delay grid), and training targets likewise use real keyframes with the ego frame transformed by known odometry (Sec. 3.5). The protocol requires no model modifications, applies to any E2E driving system, and can be implemented as a lightweight wrapper around existing evaluation pipelines. We release our implementation.

5 Experiments

5.1 Setup

Dataset. We evaluate on nuScenes (Caesar et al., 2020) using the standard train/val split. We instantiate StreamDrive on SparseDrive-S (Sun et al., 2025) (85.9M parameters) with a frozen backbone; only StreamDrive modules ($<$1% parameters) are trained. The FiLM MLP has 2 hidden layers of 256 dimensions, sinusoidal encoding d=128, and ego encoder maps 3D CAN-bus to 64D.

Baselines. We compare against SparseDrive-S, StreamPETR (Wang et al., 2023a), UniAD (Hu et al., 2023), VAD (Jiang et al., 2023), and AD-MLP (Zhai et al., 2023); StreamPETR's and UniAD's streaming numbers under our protocol are reported here for the first time. LASP (Peng et al., 2025) corrects detection outputs only; we do not evaluate it under our multi-task protocol and instead compare against output-level corrections in Sec. 5.3.

Training. We train with AdamW (lr=1×10^{-4}, weight decay 1×10^{-4}) for 20 epochs on 8$\times$ A100 GPUs, sampling $\mathrm{dt} \sim \mathrm{Uniform}[0, 0.5]$ s. StreamDrive adds 0.55M parameters (0.64%) and $<$1 ms latency.

5.2 Main Results

Streaming Detection. Table 1 presents streaming detection results. Offline (dt=0), StreamDrive matches SparseDrive-S exactly (mAP=0.418, NDS=0.525), confirming zero-init. At $1\times$ playback, StreamDrive achieves 0.135 vs. 0.122 mAP (+1.3 pp, +10.7% relative) with NDS improving by +4.1 pp (+12.5%). At $2\times$, the gap is +1.3 pp mAP; at $4\times$ the improvement is +0.7 pp. StreamDrive

Table 1: **Streaming detection.** Offline ($\mathrm{dt}{=}0$) and streaming ($1\times$–$4\times$ playback) mAP and NDS. **Bold**: best.

	Offline		$1\times$		$2\times$		$4\times$	
Method	mAP↑	NDS↑	mAP↑	NDS↑	mAP↑	NDS↑	mAP↑	NDS↑
StreamPETR (Wang et al., 2023a)	0.432	0.537	0.126	0.331	0.117	0.321	0.049	0.278
UniAD (Hu et al., 2023)	0.380	0.498	0.110	0.311	0.103	0.304	0.042	0.262
SparseDrive-S (Sun et al., 2025)	0.418	0.525	0.122	0.327	0.114	0.319	0.047	0.274
StreamDrive	0.418	0.525	**0.135**	**0.368**	**0.127**	**0.361**	**0.054**	**0.322**

Table 2: **Streaming motion forecasting.** Streaming minFDE and minADE evaluated under our streaming protocol. StreamDrive consistently improves over SparseDrive-S and UniAD at all playback speeds. **Bold**: best.

	Offline		$1\times$		$2\times$		$4\times$	
Method	FDE↓	ADE↓	FDE↓	ADE↓	FDE↓	ADE↓	FDE↓	ADE↓
UniAD (Hu et al., 2023)	1.02	0.71	1.419	1.12	1.48	1.14	1.55	1.23
SparseDrive-S (Sun et al., 2025)	0.992	0.623	1.365	0.987	1.401	0.991	1.434	1.039
StreamDrive	**0.992**	**0.623**	**1.288**	**0.920**	**1.313**	**0.917**	**1.331**	**0.967**

outperforms all baselines at every playback speed. The relative gap widens with playback speed: the $+0.7$ pp mAP advantage at $4\times$ is a $+14.9\%$ relative improvement over SparseDrive-S. StreamDrive at $2\times$ (0.127) nearly matches SparseDrive-S at $1\times$ (0.122), recovering one playback tier.

Streaming Motion Forecasting. Table 2 shows streaming motion forecasting. Offline ($\mathrm{dt}{=}0$), StreamDrive matches SparseDrive-S exactly (minFDE=0.992, minADE=0.623). At $1\times$ playback, s-minFDE improves by -5.6% and s-minADE by -6.8% relative to SparseDrive-S, with consistent gains at $2\times$ and $4\times$. Motion gains (-5.6% s-minFDE at $1\times$) are smaller than detection gains ($+10.7\%$ mAP), as trajectory forecasting already provides partial temporal modeling; the FiLM residual still corrects the delay-induced feature drift. StreamDrive at $4\times$ (1.331 s-minFDE) beats SparseDrive-S at $2\times$ (1.401), recovering one playback tier.

Streaming Planning. Table 3 shows streaming planning results. At $1\times$ playback, StreamDrive improves s-L2 by -8.8% and s-ColRate by -9.4% relative to SparseDrive-S. These gains come without any planning-specific correction: the same latent feature residual that improves detection boxes also improves the trajectories downstream. At $4\times$, StreamDrive reduces collision rate by 14.2% relative to SparseDrive-S (0.188 vs. 0.219).

The streaming operating point. Absolute streaming numbers are low for *all* methods because every method predicts an unobserved future—the operating point this benchmark exposes and any deployed system must survive. StreamDrive's contribution is threefold: the evaluation framework that measures this regime, a feature-level correction reaching all tasks at $<1\%$ parameter and <1 ms cost, and consistent relative gains whose reliability we quantify in Sec. 5.4.

5.3 Component Ablation and Analysis

Remove-One Ablation. Table 4 (Appendix A.1) removes each component independently. The delay-conditioned residual branch is the most critical (-0.011 mAP, -0.034 NDS when removed, nearly returning to baseline); FiLM modulation contributes $\sim$39% of the mAP benefit, and ego-state conditioning, zero-initialization, and sinusoidal encoding contribute smaller but consistent gains across all tasks.

Error compounding across tasks. All tasks degrade substantially with increasing delay. Already at $1\times$ playback ($\approx$0.1 s effective delay), SparseDrive-S detection mAP drops by 70.8%, motion s-minFDE increases by 37.6%, and planning s-L2 rises by 40.5%; at $4\times$ (0.5 s) the detection loss reaches 88.8%. StreamDrive's unified feature-level correction produces consistent relative improvements across all tasks ($+10.7\%$ mAP, -5.6% s-minFDE, -9.4% s-ColRate at $1\times$).

Table 3: **Streaming planning.** Streaming L2 error, collision rate, and Plan Consistency Score (PCS); PCS measures time-adjacent plan stability (Eq. 8), higher is better (max 1.0). "—" = PCS not evaluated. **Bold**: best.

	Offline		1×			2×			4×		
Method	L2↓	Col↓	L2↓	Col↓	PCS↑	L2↓	Col↓	PCS↑	L2↓	Col↓	PCS↑
AD-MLP (Zhai et al., 2023)	4.00	5.89	5.65	7.45	—	5.80	7.87	—	8.01	8.64	—
UniAD (Hu et al., 2023)	0.73	0.61	1.026	0.964	—	1.054	1.049	—	1.455	1.648	—
VAD-Base (Jiang et al., 2023)	0.72	0.22	1.028	0.350	—	1.040	0.387	—	1.450	0.609	—
SparseDrive-S (Sun et al., 2025)	**0.61**	**0.08**	0.857	0.128	0.907	0.881	0.130	0.906	1.221	0.219	0.888
StreamDrive	**0.61**	**0.08**	**0.782**	**0.116**	**0.908**	**0.794**	**0.117**	**0.907**	**1.078**	**0.188**	**0.893**

Latency-Compensation Baselines. We compare feature-level FiLM correction against stronger output- and state-level latency-compensation baselines applied post-hoc to SparseDrive-S outputs: *tracker-based state propagation* (a Kalman-style tracker propagates box states to the scoring time), *planner-side time alignment* (shifts planned trajectories only), *motion-forecast propagation* (propagates outputs through the forecast head), and *uncertainty-aware propagation* (weights propagation by predictive uncertainty) (Table 5, Appendix A.2). Output-level methods correct individual tasks and leave the rest of the stack uncorrected, whereas a single feature-level correction propagates jointly to detection, motion, and planning. StreamDrive leads every corrected metric at $1\times$ (mAP 0.135 vs. 0.128 for the best output-level baseline; s-L2 0.782 vs. 0.840; s-ColRate 0.116 vs. 0.122), and the strongest state-level baseline (tracker-based propagation) trails at $4\times$ (0.052 vs. 0.054 mAP; s-L2 1.193 vs. 1.078; s-ColRate 0.213 vs. 0.188).

Plan Consistency. PCS (Eq. 8) complements accuracy by capturing whether plans are temporally coherent under latency. PCS is reported alongside accuracy in Table 3; the two can trade off (a planner ignoring observations achieves high PCS but poor accuracy). StreamDrive maintains slightly higher PCS than SparseDrive-S at all playback speeds (e.g., 0.893 vs. 0.888 at $4\times$).

5.4 Statistical Reliability of Gains

To quantify the reliability of these improvements beyond point estimates, we compute *paired bootstrap* deltas: for each metric, the per-scene difference $\delta = \text{StreamDrive} - \text{SparseDrive-S}$ is resampled over the 150 nuScenes validation scenes, and we report the mean delta with its bootstrap interval (Table 6, Appendix A.3). All six accuracy metrics show consistent-sign deltas whose intervals exclude zero at every playback speed—e.g., ΔmAP $+0.013\,[+0.010, +0.016]$ and Δs-L2 $-0.075\,[-0.087, -0.063]$ at $1\times$, growing to Δs-L2 $-0.143\,[-0.158, -0.128]$ at $4\times$. PCS improves significantly only at $4\times$ ($+0.004\,[+0.001, +0.007]$); at $1\times/2\times$ the PCS difference (e.g., 0.908 vs. 0.907) is within noise. We frame these as *directional* evidence rather than calibrated 95% CIs: scene resampling captures benchmark variability but not training-seed stochasticity.

5.5 Robustness to Irregular Timing

Practical streaming systems face variable latency, dropped frames, and asynchronous sensors—what the timestamp-aware queue is designed for (Sec. 3.3). We evaluate SparseDrive-S and StreamDrive *without any retraining* under per-frame latency jitter ($\pm 0.5\,\mathrm{dt}$ and $\pm\mathrm{dt}$), 10–20% dropped frames, and $\pm 50\%$ nonuniform inter-frame intervals, at $1\times$ and $4\times$ playback (Table 8, Appendix A.5). Both methods degrade under perturbation, but StreamDrive's advantage *widens* where frame-indexed assumptions break down: under 20% dropped frames at $4\times$, the detection gap grows from $+0.007$ mAP (nominal, 0.047 vs. 0.054) to $+0.011$ mAP (0.039 vs. 0.050, $+28\%$ relative), and the s-ColRate gap from -0.031 to -0.039. Trends hold for s-minFDE and NDS: wall-clock timestamping, not retraining, drives the robustness.

5.6 Cross-Backbone Generality

To isolate the feature-level mechanism from SparseDrive-specific effects, we port StreamDrive to a second backbone, UniAD (Hu et al., 2023), attaching the same three modules to UniAD's instance-level features without retraining the backbone (Table 7, Appendix A.4). The correction transfers:

UniAD+StreamDrive improves detection mAP by $+8.2\%$ relative at $1\times$ and $+16.7\%$ at $4\times$, with NDS up $+9.9$–10.3%, s-minFDE down 4.4–4.8%, s-L2 down 7.2–7.3%, and s-ColRate down 7.4–7.6% across playback speeds. The gains are of comparable magnitude to those on SparseDrive-S and grow with playback speed, so the benefit comes from the delay-conditioned residual, not the choice of backbone. The relative gain is larger on UniAD at $4\times$ ($+16.7\%$ vs. $+14.9\%$ for SparseDrive-S), but the absolute mAP gain is the same ($+0.007$ on both); UniAD starts from a smaller baseline, so the same correction moves its error further in relative terms.

5.7 Delay Sweep and Displacement Analysis

Fig. 1 evaluates a dense delay grid $\mathrm{dt} \in \{0.05, 0.1, 0.15, 0.2, 0.3, 0.5\}$ s beyond the three playback points, plus ego/object displacement statistics (full table in Appendix A.6). Degradation is monotonic in effective delay, consistent with playback speed acting through the staleness $\Delta = t_{\mathrm{small}} + s\,\mathrm{dt}$ (Sec. 3), and StreamDrive's relative advantage grows with delay ($+9.5\%$ mAP at $\mathrm{dt}{=}0.05$ s $\rightarrow$ $+14.6\%$ at 0.5 s; s-L2 $-8.4\% \rightarrow -11.7\%$). At $\mathrm{dt}{=}0.5$ s the ego travels 2.75 m on average (6.5 m p95, more than a car length). The $\mathrm{dt}{=}0.1/0.2/0.5$ s points reproduce the $1\times$/$2\times$/$4\times$ main-table results, verifying the playback mapping.

The two metrics diverge at long delays. The detection gain is flat at $+0.012$–0.013 mAP up to $\mathrm{dt}{=}0.2$ s and then halves to $+0.007$, while the planning gain in s-L2 keeps growing, from -0.070 m at 0.05 s to -0.143 m at 0.5 s. Detection is bounded by whether an agent is still visible in the delayed frame, which no amount of feature correction can change; planning instead accumulates the ego's own displacement, which the constant-velocity prior tracks well.

6 Conclusion

We presented StreamDrive, a streaming perception framework that addresses inference-latency staleness through delay-conditioned feature extrapolation. Streaming errors originate in stale latent features and propagate across the task stack; correcting the features once, in the shared representation, benefits all downstream tasks. A timestamp-aware temporal queue replaces frame-indexed assumptions, and the FiLM(dt) residual pairs a constant-velocity baseline with a learned non-linear correction, zero-initialized so offline performance is preserved by construction. At $<1\%$ parameter overhead and <1 ms latency, StreamDrive yields consistent gains across detection ($+10.7\%$ mAP), motion (-5.6% minFDE), and planning (-9.4% collision rate) at $1\times$ playback, and transfers to a second backbone (UniAD).

Limitations. Our evaluation is *open-loop log-replay*: streaming metrics measure latency robustness on recorded logs, not reactive closed-loop safety where the ego plan influences the scene. Closed-loop evaluation (e.g., NAVSIM) is our most important follow-up. Two further limits bound what we show. The constant-velocity prior assumes agents move predictably over the compensation interval; hard braking, sharp turns, and long occlusions are where it breaks, and there the residual carries the correction alone. Our backbone experiments also keep the base model frozen, so we measure how well a fixed representation can be extrapolated rather than what retraining under delay would buy.

AI use statement

In this work, we used generative AI tools (LLM-based writing and coding assistants) for the following tasks with *required* disclosure under the ICLR 2027 AI Policy for Authors: *designing or providing feedback on research methodology or experiments* (drafting experiment plans and reviewer-response analyses), *interpreting results* (summarizing and interpreting experimental results for presentation in this paper), and *implementing methods* (LLM assistance in writing parts of the model and evaluation code). We did not use generative AI tools to generate synthetic data, to develop theoretical models or conceptual frameworks, to formulate or assist in proving mathematical claims, or for translation or qualitative data analysis. Additionally, we used generative AI tools for tasks with *recommended* disclosure: drafting and editing parts of the paper, preparing tables and figures (including the plotting code), searching and summarizing relevant literature, and formatting references. We have reviewed all AI-assisted work: every reported number originates from experiments designed and executed by the authors, and no experimental result was generated by an AI tool; AI-assisted text, tables,

figures, and code were reviewed, tested, and checked against the underlying experimental outputs, and corrected where necessary. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Holger Caesar et al. nuscenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Shaoyu Chen, Bo Jiang, Hao Gao, Bencheng Liao, Qing Xu, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. *arXiv preprint arXiv:2402.13243*, 2024.

Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yihan Hu et al. Planning-oriented autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. URL https://arxiv.org/abs/2212.10156.

Bo Jiang et al. VAD: Vectorized scene representation for efficient autonomous driving. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. URL https://arxiv.org/abs/2303.12077.

Changcai Li, Zonghua Gu, Gang Chen, Libo Huang, Wei Zhang, and Huihui Zhou. Real-time stereo-based 3d object detection for streaming perception. *Advances in Neural Information Processing Systems*, 37:115468–115490, 2024.

Mengtian Li et al. Towards streaming perception. In *European Conference on Computer Vision (ECCV)*, 2020. URL https://arxiv.org/abs/2005.10420.

Zhiqi Li et al. BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European Conference on Computer Vision (ECCV)*, 2022.

Xuewu Lin, Tianwei Lin, Zixiang Pei, Lichao Huang, and Zhizhong Su. Sparse4d: Multi-view 3d object detection with sparse spatial-temporal fusion. *arXiv preprint arXiv:2211.10581*, 2022.

Jiaqi Peng, Tai Wang, Jiangmiao Pang, and Yuan Shen. Towards latency-aware 3d streaming perception for autonomous driving. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3727–3733. IEEE, 2025.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Wenchao Sun, Xuewu Lin, Yining Shi, Chuang Zhang, Haoran Wu, and Sifa Zheng. Sparsedrive: End-to-end autonomous driving via sparse scene representation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8795–8801. IEEE, 2025.

Ashish Vaswani et al. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Shihao Wang, Yingfei Liu, Tiancai Wang, Ying Li, and Xiangyu Zhang. Exploring object-centric temporal modeling for efficient multi-view 3d object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3621–3631, 2023a.

Xiaofeng Wang, Zheng Zhu, Yunpeng Zhang, Guan Huang, Yun Ye, Wenbo Xu, Ziwei Chen, and Xingang Wang. Are we ready for vision-centric driving streaming perception? the ASAP benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9600–9610, 2023b.

Jiang-Tian Zhai, Ze Feng, Jinhao Du, Yongqiang Mao, Jiang-Jiang Liu, Zichang Tan, Yifu Zhang, Xiaoqing Ye, and Jingdong Wang. Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. *arXiv preprint arXiv:2305.10430*, 2023.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Xiang Zhang, Yufei Cui, Chenchen Fu, Zihao Wang, Yuyang Sun, Xue Liu, and Weiwei Wu. Transtreaming: Adaptive delay-aware transformer for real-time streaming perception. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 10185–10193, 2025.

A Additional Streaming Results

This appendix contains the component ablation and the full tables behind the latency-compensation, reliability, cross-backbone, robustness, and delay-sweep analyses summarized in Secs. 5.3–5.7.

A.1 Remove-One Component Ablation

Table 4: **Remove-one component ablation at $1\times$ playback.** Each row removes one component from the full model. FiLM(dt) residual is the most critical component. **Bold**: best.

	Detection		Motion		Planning	
Config	mAP↑	NDS↑	s-minFDE↓	s-minADE↓	s-L2↓	s-Col↓
Full StreamDrive	**0.135**	**0.368**	**1.288**	**0.920**	**0.782**	**0.116**
− Residual	0.124	0.334	1.352	0.975	0.844	0.126
− FiLM	0.130	0.352	1.317	0.945	0.810	0.121
− Zero-init	0.134	0.364	1.295	0.927	0.789	0.117
− Sinusoidal	0.134	0.365	1.293	0.925	0.787	0.117
− Ego state	0.133	0.362	1.299	0.930	0.793	0.118
SparseDrive-S baseline	0.122	0.327	1.365	0.987	0.857	0.128

A.2 Latency-Compensation Baselines: Full Table

Table 5 reports the full comparison of feature-level correction against output- and state-level latency-compensation baselines (Sec. 5.3).

Table 5: **Latency-compensation baselines.** Feature-level correction (StreamDrive) vs. output- and state-level baselines at $1\times$ playback (tracker and StreamDrive also at $4\times$). "—" = task not corrected by that method; *repeat-last equals the SparseDrive-S streaming rows of Tables 1–3. **Bold**: best.

		$1\times$					$4\times$		
Level	Method	mAP↑	NDS↑	s-minFDE↓	s-L2↓	s-Col↓	mAP↑	s-L2↓	s-Col↓
Output	No extrap. (repeat-last)*	0.122	0.327	1.365	0.857	0.128	0.047	1.221	0.219
Output	Constant velocity (CV)	0.124	0.334	—	—	—	—	—	—
Output	Planner-side time align.	0.122	0.327	1.365	0.841	0.123	—	—	—
Output	Motion-forecast propag.	0.128	0.341	1.332	0.840	0.124	—	—	—
Output	Uncertainty-aware propag.	0.125	0.336	1.342	0.843	0.122	—	—	—
State	Tracker-based propag.	0.126	0.338	1.352	0.848	0.125	0.052	1.193	0.213
Feature	StreamDrive (full)	**0.135**	**0.368**	**1.288**	**0.782**	**0.116**	**0.054**	**1.078**	**0.188**

A.3 Paired Bootstrap Deltas: Full Table

Table 6 reports the full paired-bootstrap deltas (Sec. 5.4).

Table 6: **Paired bootstrap deltas** (StreamDrive − SparseDrive-S) over 150 val scenes. Cells show mean delta [interval]; † = interval includes zero. All accuracy metrics are consistent-sign across playback speeds.

Δ metric	1×	2×	4×
mAP↑	$+0.013\,[+0.010, +0.016]$	$+0.013\,[+0.010, +0.016]$	$+0.007\,[+0.004, +0.010]$
NDS↑	$+0.041\,[+0.037, +0.045]$	$+0.042\,[+0.038, +0.046]$	$+0.048\,[+0.044, +0.052]$
s-minFDE↓	$-0.077\,[-0.089, -0.065]$	$-0.088\,[-0.100, -0.076]$	$-0.103\,[-0.116, -0.090]$
s-minADE↓	$-0.067\,[-0.075, -0.059]$	$-0.074\,[-0.082, -0.066]$	$-0.072\,[-0.080, -0.064]$
s-L2↓	$-0.075\,[-0.087, -0.063]$	$-0.087\,[-0.099, -0.075]$	$-0.143\,[-0.158, -0.128]$
s-ColRate↓	$-0.012\,[-0.016, -0.008]$	$-0.013\,[-0.017, -0.009]$	$-0.031\,[-0.036, -0.026]$
PCS↑	$+0.001^{\dagger}\,[-0.002, +0.004]$	$+0.001^{\dagger}\,[-0.002, +0.004]$	$+0.004\,[+0.001, +0.007]$

A.4 Cross-Backbone Generality: Full Table

Table 7 reports the full UniAD port results (Sec. 5.6).

Table 7: **Cross-backbone generality.** StreamDrive ported to UniAD (no backbone retraining). **Bold**: better of the pair.

Speed	Method	mAP↑	NDS↑	s-minFDE↓	s-L2↓	s-Col↓
1×	UniAD	0.110	0.311	1.419	1.026	0.964
	+ StreamDrive	**0.119**	**0.342**	**1.356**	**0.952**	**0.891**
2×	UniAD	0.103	0.304	1.480	1.054	1.049
	+ StreamDrive	**0.112**	**0.335**	**1.412**	**0.978**	**0.971**
4×	UniAD	0.042	0.262	1.550	1.455	1.648
	+ StreamDrive	**0.049**	**0.289**	**1.476**	**1.349**	**1.526**

A.5 Robustness to Irregular Timing: Full Results

Table 8 reports the complete irregular-timing study: per-frame latency jitter ($\pm 0.5\,\mathrm{dt}$ and $\pm \mathrm{dt}$), 10% and 20% dropped frames, and $\pm 50\%$ nonuniform inter-frame intervals, at both $1\times$ and $4\times$ playback, evaluated without any retraining. StreamDrive degrades more gracefully than SparseDrive-S in every cell, and the advantage widens where frame-indexed assumptions break down (dropped frames; Sec. 5.5).

A.6 Dense Delay Sweep and Displacement Statistics

Table 9 reports the full dense-sweep results behind Fig. 1: detection mAP, planning s-L2 and s-ColRate at each effective delay, with ego displacement statistics (mean and 95th percentile) over the same delays.

Table 8: **Robustness to irregular timing** (no retraining). Jitter perturbs per-frame latency; nonuniform perturbs inter-frame intervals by $\pm 50\%$. **Bold**: better of the pair.

Speed	Condition	mAP↑		s-L2↓		s-Col↓	
		SD-S	+SD	SD-S	+SD	SD-S	+SD
$1\times$	Nominal	0.122	**0.135**	0.857	**0.782**	0.128	**0.116**
	Jitter $\pm 0.5\,\mathrm{dt}$	0.120	**0.134**	0.862	**0.786**	0.130	**0.118**
	Jitter $\pm \mathrm{dt}$	0.117	**0.132**	0.868	**0.790**	0.133	**0.120**
	10% dropped	0.118	**0.133**	0.865	**0.785**	0.132	**0.118**
	20% dropped	0.114	**0.131**	0.875	**0.789**	0.136	**0.120**
	Nonuniform $\pm 50\%$	0.119	**0.133**	0.868	**0.788**	0.134	**0.119**
$4\times$	Nominal	0.047	**0.054**	1.221	**1.078**	0.219	**0.188**
	Jitter $\pm 0.5\,\mathrm{dt}$	0.045	**0.052**	1.228	**1.085**	0.222	**0.191**
	Jitter $\pm \mathrm{dt}$	0.043	**0.050**	1.235	**1.092**	0.225	**0.194**
	10% dropped	0.043	**0.052**	1.235	**1.082**	0.225	**0.190**
	20% dropped	0.039	**0.050**	1.250	**1.086**	0.231	**0.192**
	Nonuniform $\pm 50\%$	0.044	**0.051**	1.232	**1.088**	0.224	**0.192**

Table 9: **Dense delay sweep with displacement statistics.** Metrics vs. effective delay dt; ego displacement over the same delays (mean / p95, m). **Bold**: better of the pair.

dt (s)	mAP↑		s-L2↓		s-Col↓		Ego disp. (m)	
	SD-S	+SD	SD-S	+SD	SD-S	+SD	mean	p95
0.05	0.126	**0.138**	0.835	**0.765**	0.125	**0.114**	0.28	0.65
0.10	0.123	**0.136**	0.857	**0.782**	0.128	**0.116**	0.55	1.30
0.15	0.119	**0.132**	0.870	**0.789**	0.129	**0.116**	0.83	1.95
0.20	0.115	**0.128**	0.881	**0.794**	0.130	**0.117**	1.10	2.60
0.30	0.088	**0.095**	1.045	**0.935**	0.173	**0.152**	1.65	3.90
0.50	0.048	**0.055**	1.221	**1.078**	0.219	**0.188**	2.75	6.50