TwinFlux: One-Step Discrete-Continuous Flow for End-to-End Autonomous Driving

Ziming Zhu* **Zheng Shen*** **Qin Guo** **Yujia Zhang** **Yu Zhu†**
School of Information Science and Engineering
East China University of Science and Technology
{zimingzhu, y30251003, y10250156, 23013019}@mail.ecust.edu.cn
zhuyu@ecust.edu.cn

Abstract

End-to-end autonomous driving planning requires trajectories that are multimodal, scene-compliant, and efficient to generate. Deterministic regression is fast but often collapses to a single future, while trajectory-vocabulary methods depend on predefined candidate coverage and iterative generative planners incur multi-step inference cost. We propose TwinFlux, a one-step discrete-continuous trajectory generation framework based on SplitMeanFlow. TwinFlux starts from data-driven trajectory prototypes and refines them through parallel heatmap and offset branches. The heatmap branch is designed to provide topology-aware coarse localization on a BEV grid, while the offset branch predicts grid-relative geometric refinement. A differentiable trajectory decoupling-and-reconstruction module connects this representation with physical trajectories. During training, TwinFlux enforces piecewise displacement consistency in reconstructed trajectory space, aligning direct one-step prediction with accumulated sub-interval evolution without Jacobian-vector-product computation or inference overhead. Experiments on NAVSIMv1 and NAVSIMv2 show that TwinFlux outperforms recent end-to-end planning baselines while maintaining real-time inference. Project repository: `https://github.com/ECUST-MMAI/TwinFlux`.

1 Introduction

End-to-end trajectory planning maps multi-sensor observations directly to executable trajectories, reducing error propagation across modular perception, prediction, and planning pipelines. Recent planners such as TCP [1], InterFuser [2], PlanT [3], and GenAD [4] further integrate perception, prediction, control, and planning. However, urban driving is inherently multimodal: unprotected turns, intersection negotiation, and obstacle avoidance often admit multiple feasible behaviors. An effective planner must therefore generate diverse trajectories while maintaining scene compliance and real-time inference.

Existing planners usually fall into three paradigms. Deterministic planners such as TransFuser [5], UniAD [6], and VAD [7] are stable but often regress a single future. Trajectory-vocabulary planners such as VADv2 [8] improve multimodality by scoring predefined candidates, but their performance depends on candidate coverage and vocabulary quality. Anchor-refinement methods such as Hydra-MDP [9] further adjust candidate trajectories, yet still rely on predefined source structures. Diffusion planners such as DiffusionDrive [10], ResAD [11], and Diffusion Planner [12] provide flexible generation, building on diffusion models [13, 14], but iterative denoising introduces non-negligible latency.

*These authors contributed equally to this work.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

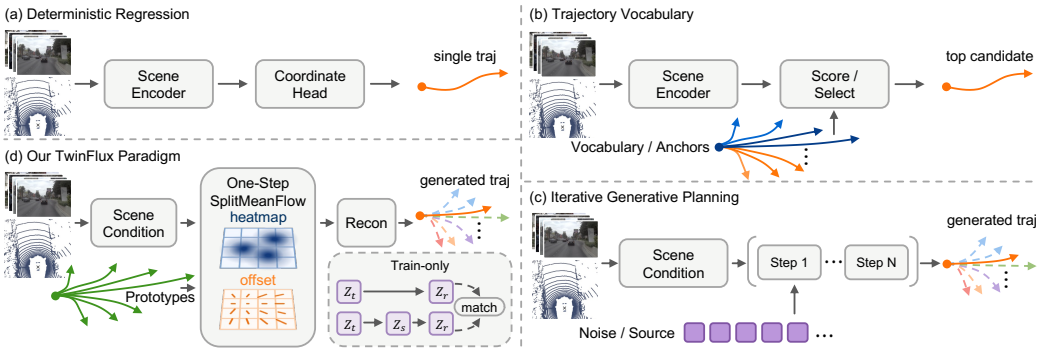


Figure 1: Paradigm comparison for end-to-end trajectory planning. (a) Deterministic regression predicts one trajectory directly. (b) Trajectory-vocabulary methods score predefined anchors. (c) Iterative generative planners refine samples over multiple steps. (d) TwinFlux performs one-step heatmap-offset refinement from prototypes, with training-only consistency supervision.

Figure 1 summarizes these paradigm-level differences. These limitations reveal a trade-off among multimodal coverage, scene compliance, and inference efficiency. Efficient transport models such as Rectified Flow [15] and SplitMeanFlow [16] offer a promising alternative for fast one-step generation. Nevertheless, directly applying flow generation to continuous trajectory coordinates raises representation concerns: a single coordinate-space representation lacks explicit road-topology constraints, may produce spatially non-compliant trajectories, and can be sensitive to temporal coordinate-scale imbalance because farther future waypoints naturally have larger magnitudes than near-term waypoints.

To address this problem, we propose TwinFlux, a SplitMeanFlow-based discrete-continuous parallel framework for end-to-end multimodal planning. Instead of generating trajectories through a single continuous coordinate-space representation, TwinFlux decouples generation into a discrete heatmap branch and a continuous offset branch. The discrete heatmap branch performs average-flow generation on a BEV-aligned grid for topology-aware coarse localization, while the continuous offset branch predicts grid-relative coordinate and heading refinements for geometric precision without relying solely on absolute coordinate regression. The two branches share scene-conditioned BEV features, are trained jointly, and run fully in parallel during inference. TwinFlux further introduces trajectory decoupling and reconstruction to connect physical trajectories with the heatmap-offset representation, and uses piecewise displacement consistency to align one-step direct generation with accumulated-flow evolution in reconstructed trajectory space.

Our contributions are summarized as follows:

- We propose TwinFlux, a discrete-continuous parallel end-to-end planning framework that decouples multimodal trajectory generation into discrete topology-aware heatmap prediction and continuous offset refinement, balancing scene compliance, geometric precision, and real-time efficiency.
- We adapt SplitMeanFlow to autonomous driving trajectory planning and introduce training-time piecewise displacement consistency in reconstructed trajectory space, enabling high-quality one-step generation without iterative denoising.
- We design hierarchical conditional cross-attention, dual-timestep affine modulation, and reversible trajectory decoupling-reconstruction to improve scene-conditioned generation, temporal modeling, and physical trajectory recovery.
- Extensive experiments on NAVSIMv1 and NAVSIMv2 show that TwinFlux achieves competitive planning performance and real-time inference efficiency, outperforming representative deterministic, vocabulary-based, diffusion-based, and flow-based planners.

2 Related Work

We review related work from three perspectives: end-to-end trajectory planning, flow-based generative planning, and discrete-continuous trajectory generation.

2.1 End-to-End Autonomous Driving Trajectory Planning

End-to-end planning maps sensor observations directly to executable trajectories. BEV- and Transformer-based planners including TransFuser [5], UniAD [6], VAD [7], ST-P3 [17], and NEAT [18] improved planning with multimodal feature fusion and joint perception-planning optimization. However, deterministic regression usually provides limited coverage of multimodal urban behaviors. Anchor- or vocabulary-based methods, such as MP3 [19], Hydra-MDP [9], VADv2 [8], and LaPred [20], together with target-driven prediction methods including MultiPath [21], TNT [22], and DenseTNT [23], improve multimodal coverage by discretizing candidate futures, but introduce coverage, scoring, and refinement trade-offs. DrivoR [24] uses compact camera-aware registers and separate trajectory generation and oracle-supervised scoring decoders, highlighting the role of candidate ranking. DiffusionDrive [10], ResAD [11], and Diffusion Planner [12] provide flexible generation but require iterative denoising. TwinFlux targets one-step multimodal generation using a BEV heatmap-offset representation.

2.2 Flow-Based Generative Planning

Flow matching learns a vector field that transports a source distribution to the data distribution, with variants such as Conditional Flow Matching [25]. MeanFlow [26] enables one-step generation through average velocity-field learning, and SplitMeanFlow [16] removes Jacobian-vector-product training with piecewise displacement consistency. Recent driving works include TrajFlow [27] for behavior-sequence modeling, GoalFlow [28] for goal-guided planning, MoFlow [29] for K-shot flow generation and distillation, FlowS [30] for local transport from plausible futures, MeanFuser [31] for Gaussian-mixture MeanFlow planning, FlowDrive [32] for long-tail robust rectified flow, and Flow Planner [33] for tokenized interactive planning. WAM-Flow [34] combines metric-aligned discrete tokens with a 1.5B vision-language-action backbone and simulator-guided GRPO, reporting 89.1 PDMS at one step and 90.3 at five steps on NAVSIMv1. These methods demonstrate the promise of flow-based planning, but commonly operate in a single continuous trajectory space or rely on stochastic sampling, goals, guidance, tokenization, or post-processing. TwinFlux instead combines SplitMeanFlow with fixed cluster-prototype sources and a parallel heatmap-offset representation, turning multimodal generation into scene-conditioned refinement from data-driven trajectory modes.

2.3 Discrete-Continuous Hybrid Trajectory Generation

Discrete-continuous hybrids combine coarse discrete localization with continuous refinement. In autonomous driving, M2I [35] and PnPNet [36] constrain candidate search before refinement, while Hydra-MDP [9] and VADv2 [8] pair discrete anchors or trajectory vocabularies with continuous offsets. TwinFlux differs by using parallel discrete heatmap and continuous offset branches under SplitMeanFlow. Clustered prototypes serve only as fixed source endpoints, not final vocabulary predictions, enabling one-step multimodal generation without stochastic intra-mode sampling.

3 Method

3.1 Problem Formulation

We formulate end-to-end trajectory planning as learning a mapping from multimodal observations $X_{\text{obs}} = \{\mathbf{I}, \mathbf{B}_{\text{lidar}}, s\}$ to a future trajectory $\hat{\tau} = \{\hat{\tau}_i\}_{i=1}^{T}$, where $\mathbf{I}$, $\mathbf{B}_{\text{lidar}}$, and s denote camera images, rasterized BEV LiDAR input, and the ego state, respectively. Each future state is $\hat{\tau}_i = [\hat{x}_i, \hat{y}_i, \hat{\theta}_i]$, where $(\hat{x}_i, \hat{y}_i)$ and $\hat{\theta}_i$ are the ego-centric position and heading. During training, expert trajectory $\tau^* = \{\tau_i^*\}_{i=1}^{T}$ is used for imitation learning. With learnable parameters Θ, the model is written as $\hat{\tau} = f(X_{\text{obs}}; \Theta)$.

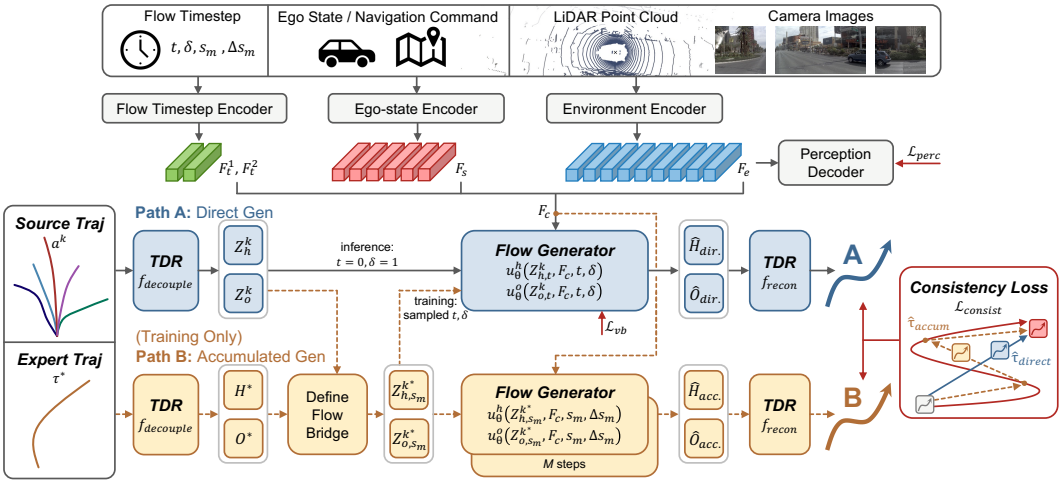


Figure 2: Overall architecture of TwinFlux. Path A performs one-step direct generation and is used during inference, while Path B is introduced only during training to enforce accumulated-flow consistency in reconstructed trajectory space.

3.2 SplitMeanFlow for One-Step Trajectory Generation

TwinFlux adopts SplitMeanFlow [16] to support one-step generative trajectory planning. Given an interval $0 \le t < r \le 1$, a partition $t = s_0 < s_1 < \cdots < s_M = r$, and $\Delta s_m = s_{m+1} - s_m$, SplitMeanFlow enforces piecewise displacement consistency:

$$(r-t)u_\theta(z_t, r, t) = \sum_{m=0}^{M-1} \Delta s_m \, u_\theta(z_{s_m}, s_{m+1}, s_m). \tag{1}$$

This Jacobian-free constraint preserves one-step generation while providing the basis for the training-time accumulated-flow consistency used in TwinFlux. Additional background on flow matching [37], MeanFlow [26], and SplitMeanFlow [16] is provided in Appendix A.

3.3 Overview of TwinFlux

TwinFlux is a discrete-continuous parallel architecture for end-to-end multimodal trajectory planning. As shown in Fig. 2, it separates scene-conditioned feature encoding, one-step direct generation, and training-time accumulated-flow consistency. BEV scene features are computed on a feature grid, while heatmap-offset generation and Trajectory Decoupling and Reconstruction (TDR) are defined on a separate trajectory generation grid.

Scene-conditioned feature encoding (top feature-encoding block in Fig. 2). The environment encoder fuses camera and rasterized BEV LiDAR inputs into BEV scene features F_e, the ego-state encoder embeds vehicle states and navigation commands into F_s, and the flow-timestep encoder provides temporal conditioning. These features form the shared conditions F_c for both heatmap and offset generation. Scene-encoding details are provided in Appendix B.

One-step direct generation (Path A in Fig. 2). Given a fixed data-driven trajectory prototype, TDR converts it into heatmap-offset source fields. The discrete-continuous generator predicts heatmap and offset updates in parallel, and TDR reconstructs the resulting physical trajectory. This direct path is the only path used during inference.

Training-time accumulated-flow consistency (Path B in Fig. 2). During training, the selected prototype source and the expert trajectory define an expert-anchored flow bridge. Path A predicts a sampled interval in one step, while Path B accumulates predictions over subintervals on the same bridge. Their reconstructed trajectories are aligned in physical trajectory space, and a velocity-boundary term further anchors the local bridge velocity.

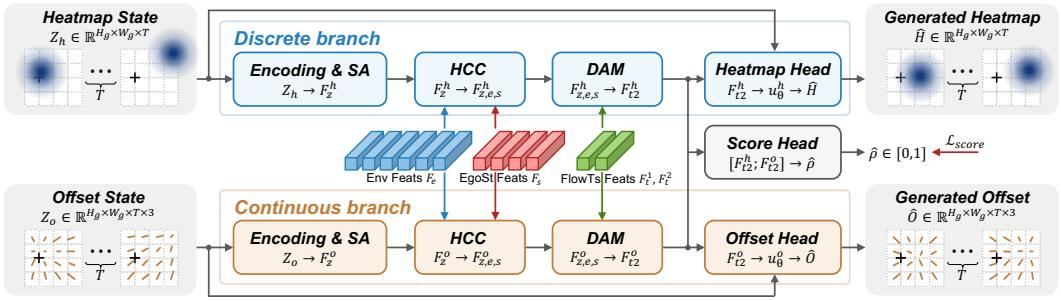


Figure 3: Discrete-continuous parallel generator, expanding the generator block in Fig. 2. The discrete and continuous branches predict BEV heatmaps and grid-relative offsets, respectively, while a score head predicts candidate confidence for top-1 selection.

3.4 Trajectory Decoupling and Reconstruction

TwinFlux uses TDR to bridge physical trajectory space and the heatmap-offset representation. The decoupling function maps a physical trajectory into a normalized heatmap and a grid-relative offset field,

$$(H, O) = f_{\text{decouple}}(\tau), \qquad H \in \mathbb{R}^{H_g \times W_g \times T}, \quad O \in \mathbb{R}^{H_g \times W_g \times T \times 3}. \tag{2}$$

For trajectory state $\tau_i = [x_i, y_i, \theta_i]$ and grid anchor $\mathbf{g}_{uv} = [g_x(u,v), g_y(u,v)]$, each heatmap slice H_i is a normalized Gaussian probability map and each offset vector stores an exact residual:

$$H_i(u,v) = \text{Softmax}_{u,v}\left(-\frac{\|\mathbf{g}_{uv} - [x_i, y_i]\|_2^2}{2\sigma^2}\right), \qquad O_i(u,v) = \big[x_i - g_x(u,v),\ y_i - g_y(u,v),\ \theta_i\big]. \tag{3}$$

Given predicted fields $(\hat{H}, \hat{O})$, reconstruction normalizes each heatmap slice as $\hat{P}_i = \text{Normalize}(\hat{H}_i)$ with $\sum_{u,v} \hat{P}_i(u,v) = 1$. Writing $\hat{O}_i(u,v) = [\hat{O}_i^x(u,v), \hat{O}_i^y(u,v), \hat{O}_i^\theta(u,v)]$, the recovered trajectory state is

$$\begin{aligned} \hat{x}_i &= \sum_{u,v} \hat{P}_i(u,v)\big(g_x(u,v) + \hat{O}_i^x(u,v)\big), \\ \hat{y}_i &= \sum_{u,v} \hat{P}_i(u,v)\big(g_y(u,v) + \hat{O}_i^y(u,v)\big), \\ \hat{\theta}_i &= \sum_{u,v} \hat{P}_i(u,v)\hat{O}_i^\theta(u,v). \end{aligned} \tag{4}$$

Thus, $\hat{\tau} = f_{\text{recon}}(\hat{H}, \hat{O})$. Because the residuals are defined globally relative to every grid anchor, $f_{\text{recon}}(f_{\text{decouple}}(\tau)) = \tau$ while keeping reconstruction differentiable. TDR is designed to mitigate temporal coordinate-scale imbalance by representing coarse localization as a normalized spatial distribution and geometry as grid-relative residuals. Grid definitions and reversibility analysis are provided in Appendix D; empirical heatmap controls are in Appendix G.4.

3.5 Discrete-Continuous Parallel Generator

With the heatmap-offset representation defined above, the discrete-continuous generator contains a discrete heatmap branch and a continuous offset branch, as shown in Fig. 3. Given an input state (Z_h, Z_o), which is a fixed prototype source at inference or an intermediate bridge state during training, the heatmap branch predicts coarse topology-aware localization while the offset branch predicts continuous geometric refinement. Both branches tokenize the grid-structured input over future steps, use Hierarchical Conditional Cross-Attention (HCC) to inject compact scene tokens and ego-state features, and use Dual-Timestep Affine Modulation (DAM) to condition on the flow start time t and interval $\Delta t = r - t$.

The modulated branch features are decoded into grid-structured velocity fields $u_\theta^h(Z_h, F_c, t, \Delta t)$ and $u_\theta^o(Z_o, F_c, t, \Delta t)$, and a lightweight score head outputs candidate confidence $\hat{\rho} \in [0,1]$. For $\Delta t > 0$, the heatmap and offset outputs are interpreted as average velocities:

$$z_r = z_t + (r - t)\, u_\theta(z_t, F_c, t, \Delta t). \tag{5}$$

For $\Delta t = 0$, used only by the velocity-boundary loss in Sec. 3.6, the output is interpreted as an instantaneous velocity estimate. At inference, setting $t = 0$ and $r = 1$ gives one-step generation for each prototype source k:

$$\hat{H}^k = Z_h^k + u_\theta^h(Z_h^k, F_c, 0, 1), \qquad \hat{O}^k = Z_o^k + u_\theta^o(Z_o^k, F_c, 0, 1), \tag{6}$$

$$\hat{\tau}^k = f_{\text{recon}}(\hat{H}^k, \hat{O}^k), \qquad k_{\text{top}} = \arg\max_{k \in \{1,\dots,K\}} \hat{\rho}_k, \qquad \hat{\tau}_{\text{top}} = \hat{\tau}^{k_{\text{top}}}. \tag{7}$$

Additional implementation details of self-attention, HCC, DAM, and trajectory scoring are given in Appendix C.

3.6 Training Objective

TwinFlux is trained with imitation learning. Expert trajectories supervise the generator through component assignment, endpoint-paired bridge construction, bridge-velocity supervision, and candidate scoring, rather than through direct point-wise trajectory-coordinate regression. The training objective combines multimodal component matching, trajectory-space flow consistency, velocity-boundary supervision, trajectory scoring, and auxiliary perception supervision.

Multimodal component matching. Let $\mathcal{A} = \{a^k\}_{k=1}^K$ be K data-driven trajectory prototypes obtained by clustering expert demonstrations. Given expert trajectory $\tau^* = \{\tau_i^*\}_{i=1}^T$, we select the nearest prototype directly in physical trajectory space:

$$d_k = \sum_{i=1}^{T} \|a_i^k - \tau_i^*\|_2, \qquad k^* = \arg\min_{k \in \{1,\dots,K\}} d_k. \tag{8}$$

Only k^* contributes to the flow loss computation; the score loss below supervises all components. This best-mode assignment is what prevents all modes from being pulled toward the same expert trajectory for each sample, and encourages different components to specialize to different driving modes.

Trajectory scoring. The same best-mode assignment provides binary labels $y_k = \mathbb{I}(k = k^*)$ for the score head. The direct path is evaluated for all K prototype-conditioned candidates, and the score loss $\mathcal{L}_{\text{score}}$ trains the model to rank the nearest expert-matched component as the top-1 trajectory. The selected component is then refined by the flow loss below; scoring details are given in Appendix C.

Trajectory-space consistency. For the selected component, the prototype source and the expert trajectory define the two endpoints of the flow bridge. We decouple both endpoints into their heatmap-offset representations,

$$(Z_h^{k^*}, Z_o^{k^*}) = f_{\text{decouple}}(a^{k^*}), \qquad (H^*, O^*) = f_{\text{decouple}}(\tau^*), \tag{9}$$

without adding stochastic intra-mode perturbations to the prototype source. The bridge state at any flow time $s \in [0, 1]$ is then defined between the selected prototype endpoint $(Z_h^{k^*}, Z_o^{k^*})$ and the expert endpoint (H^*, O^*):

$$Z_{h,s}^{k^*} = (1 - s)Z_h^{k^*} + sH^*, \qquad Z_{o,s}^{k^*} = (1 - s)Z_o^{k^*} + sO^*. \tag{10}$$

During training, we randomly sample an interval $0 \le t < r \le 1$ and set $\delta = r - t$. For the selected component, the one-step predictions over this interval are

$$\hat{H}_{\text{direct}}^{k^*,t,r} = Z_{h,t}^{k^*} + \delta\, u_\theta^h(Z_{h,t}^{k^*}, F_c, t, \delta), \qquad \hat{O}_{\text{direct}}^{k^*,t,r} = Z_{o,t}^{k^*} + \delta\, u_\theta^o(Z_{o,t}^{k^*}, F_c, t, \delta). \tag{11}$$

For the accumulated path, we partition the sampled interval as $t = s_0 < s_1 < \cdots < s_M = r$ with $\Delta s_m = s_{m+1} - s_m$. The accumulated predictions are

$$\hat{H}_{\text{accum}}^{k^*,t,r} = Z_{h,t}^{k^*} + \sum_{m=0}^{M-1} \Delta s_m u_\theta^h(Z_{h,s_m}^{k^*}, F_c, s_m, \Delta s_m), \tag{12}$$

$$\hat{O}_{\text{accum}}^{k^*,t,r} = Z_{o,t}^{k^*} + \sum_{m=0}^{M-1} \Delta s_m u_\theta^o(Z_{o,s_m}^{k^*}, F_c, s_m, \Delta s_m). \tag{13}$$

After reconstruction,

$$\hat{\tau}^{k^*,t,r}_{\text{direct}} = f_{\text{recon}}(\hat{H}^{k^*,t,r}_{\text{direct}}, \hat{O}^{k^*,t,r}_{\text{direct}}), \qquad \hat{\tau}^{k^*,t,r}_{\text{accum}} = f_{\text{recon}}(\hat{H}^{k^*,t,r}_{\text{accum}}, \hat{O}^{k^*,t,r}_{\text{accum}}). \tag{14}$$

The consistency loss is defined in physical trajectory space, using the accumulated trajectory as a stop-gradient target:

$$\mathcal{L}_{\text{consist}} = \mathbb{E}_{t,r}\left[\left\|\hat{\tau}^{k^*,t,r}_{\text{direct}} - \text{sg}\left(\hat{\tau}^{k^*,t,r}_{\text{accum}}\right)\right\|_2^2\right]. \tag{15}$$

Here $\text{sg}(\cdot)$ denotes the stop-gradient operation. The expert trajectory provides supervision as the data endpoint of the flow bridge, while the flow losses below train the selected component to learn a prototype-to-data generative mapping. Because multimodality is anchored by cluster prototypes, the generator performs scene-conditioned refinement from each prototype source rather than discovering behavioral modes from random noise.

Instantaneous velocity-boundary supervision. Interval consistency provides self-consistency, but does not directly supervise the local bridge velocity. For samples selected by $b_{\text{vb}} \sim \text{Bernoulli}(p_{\text{vb}})$, we replace interval consistency with a velocity-boundary loss. At sampled flow time q, the generator receives $(Z^{k^*}_{h,q}, Z^{k^*}_{o,q})$ with $\Delta t = 0$ and predicts the instantaneous velocity of the linear bridge:

$$v^h_\theta(q) = u^h_\theta(Z^{k^*}_{h,q}, F_c, q, 0), \quad v^o_\theta(q) = u^o_\theta(Z^{k^*}_{o,q}, F_c, q, 0), \quad V^{k^*}_h = H^* - Z^{k^*}_h, \quad V^{k^*}_o = O^* - Z^{k^*}_o. \tag{16}$$

The boundary and flow losses are

$$\mathcal{L}_{\text{vb}} = \mathbb{E}_q\left[\|v^h_\theta(q) - V^{k^*}_h\|_2^2 + \|v^o_\theta(q) - V^{k^*}_o\|_2^2\right]. \tag{17}$$

$$\mathcal{L}_{\text{flow}} = (1 - b_{\text{vb}})\mathcal{L}_{\text{consist}} + b_{\text{vb}}\lambda_{\text{vb}}\mathcal{L}_{\text{vb}}, \tag{18}$$

where λ_{vb} balances the velocity-boundary term.

Auxiliary perception supervision. An auxiliary perception decoder is applied to F_e during training for object detection and lane-map prediction, yielding $\mathcal{L}_{\text{perc}}$ as detailed in Appendix E. The final objective is

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{perc}} + \beta\mathcal{L}_{\text{flow}} + \eta\mathcal{L}_{\text{score}}. \tag{19}$$

Here α, β, and η balance the perception, flow, and score losses. Inference uses only the direct path and score head, introducing no accumulated-flow overhead.

4 Experiments

4.1 Datasets and Evaluation Metrics

We evaluate TwinFlux on the public navtest splits of NAVSIMv1 and NAVSIMv2 [38, 39], using devkits v1.1 and v2.2, respectively. Both evaluations use non-reactive traffic; NAVSIMv2 uses the one-stage evaluator. We report PDMS on NAVSIMv1 and EPDMS on NAVSIMv2, aggregating safety, rule compliance, progress, and comfort-related simulation metrics. We additionally evaluate reactive closed-loop planning on Bench2Drive [40]. Detailed protocols and metric definitions are provided in Appendix F.

4.2 Implementation Details

For NAVSIM, the camera input consists of cropped and downsampled front-left, front, and front-right images concatenated into a 1024×256 image. We also use rasterized BEV LiDAR input centered on the ego vehicle. The BEV feature grid is 256×256 over a $64\text{m} \times 64\text{m}$ ego-centric region, and the trajectory generation grid is 32×32. We plan $T = 8$ steps over 4 seconds, use $K = 10$ K-means trajectory prototypes as fixed source endpoints without Gaussian jitter, and set the velocity-boundary sampling probability to $p_{\text{vb}} = 0.7$. Models are trained for 100 epochs with AdamW [41] on 8 RTX 3090 GPUs; inference generates 10 trajectories in parallel and selects the highest-scoring one on a single RTX 3090. Additional architecture, optimizer, and latency details are in Appendix F.

Table 1: Test performance on the NAVSIMv1 dataset. A dash indicates an unreported metric.

Method	Venue	Input	Img. Backbone	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
UniAD [6]	CVPR'23	C	ResNet-34	97.8	91.9	92.9	**100.0**	78.8	83.4
PARA-Drive [42]	CVPR'24	C	ResNet-34	97.9	92.4	93.0	99.8	79.3	84.0
World4Drive [43]	ICCV'25	C	ResNet-34	97.4	94.3	92.8	**100.0**	79.9	85.1
Epona [44]	ICCV'25	C	Transformer	97.9	95.1	93.8	99.9	80.4	86.2
Hydra-MDP++ [45]	arXiv'25	C	ResNet-34	97.6	96.0	93.1	**100.0**	80.4	86.6
MeanFuser [31]	CVPR'26	C	ResNet-34	98.6	97.0	95.0	**100.0**	82.8	89.0
VADv2 [8]	ICLR'26	C	ResNet-34	98.3	97.4	95.7	**100.0**	82.3	89.3
TransFuser [5]	TPAMI'23	C & L	ResNet-34	97.7	92.8	92.8	**100.0**	79.2	84.0
DRAMA [46]	arXiv'24	C & L	ResNet-34	98.0	93.1	94.8	**100.0**	80.1	85.5
GoalFlow [28]	CVPR'25	C & L	ResNet-34	98.3	93.8	94.3	**100.0**	79.8	85.7
Hydra-MDP [9]	arXiv'24	C & L	ResNet-34	98.3	96.0	94.6	**100.0**	78.7	86.5
DiffusionDrive [10]	CVPR'25	C & L	ResNet-34	98.2	96.2	94.7	**100.0**	82.2	88.1
WoTE [47]	ICCV'25	C & L	ResNet-34	98.5	96.8	94.9	99.9	81.9	88.3
Hydra-NeXt [48]	ICCV'25	C & L	ResNet-34	98.1	**97.7**	94.6	**100.0**	81.8	88.6
GaussianFusion [49]	NeurIPS'25	C & L	ResNet-34	98.3	97.2	94.6	–	83.0	88.8
DiffE2E [50]	NeurIPS'25	C & L	ResNet-34	**99.2**	96.8	96.7	**100.0**	83.6	89.8
TwinFlux	Ours	C & L	ResNet-34	99.1	97.5	**96.9**	**100.0**	**83.7**	**90.1**

Table 2: Test performance on the NAVSIMv2 dataset.

Method	Venue	NC ↑	DAC ↑	DDC ↑	TLC ↑	EP ↑	TTC ↑	LK ↑	HC ↑	EC ↑	EPDMS ↑
Ego Status MLP [38]	NeurIPS'24	93.1	77.9	92.7	99.6	86.0	91.5	89.4	**98.3**	85.4	64.0
TransFuser [5]	TPAMI'23	96.9	89.9	97.8	99.7	87.1	95.4	92.7	**98.3**	87.2	76.7
Hydra-MDP++ [45]	arXiv'25	97.2	97.5	99.4	99.6	83.1	96.5	94.4	98.2	70.9	81.4
DriveSuprim [51]	AAAI'26	97.5	96.5	99.4	99.6	88.4	96.6	95.5	**98.3**	77.0	83.1
ARTEMIS [52]	RAL'26	98.3	95.1	98.6	**99.8**	87.6	97.4	96.5	**98.3**	89.1	83.1
PRIX [53]	RAL'26	98.0	95.6	99.5	**99.8**	87.4	97.2	97.1	**98.3**	87.6	84.2
GaussianFusion [49]	NeurIPS'25	98.3	97.3	99.0	–	87.5	97.4	97.4	–	–	85.0
VADv2 [8]	ICLR'26	98.0	**98.3**	99.4	**99.8**	87.1	96.8	95.2	**98.3**	88.1	85.8
DiffusionDrive [10]	CVPR'25	98.2	96.3	99.4	**99.8**	87.4	97.4	97.0	**98.3**	87.7	88.3
MeanFuser [31]	CVPR'26	98.3	97.2	**99.6**	**99.8**	87.6	97.4	97.3	**98.3**	88.2	89.5
TwinFlux	Ours	**98.5**	97.9	**99.6**	**99.8**	**88.5**	**97.8**	**98.1**	**98.3**	**89.3**	**90.5**

4.3 Main Results

We compare TwinFlux with representative end-to-end planning methods on NAVSIMv1 and NAVSIMv2, covering deterministic, trajectory-vocabulary-based, diffusion-based, and flow-based planning paradigms. The tables use the public-result snapshot of April 24, 2026; baseline values are paper- or leaderboard-reported unless marked as reproduced (Appendix F).

Results on NAVSIMv1. Table 1 reports the non-reactive simulation results on NAVSIMv1. TwinFlux achieves the best overall performance with a PDMS of 90.1, outperforming DiffE2E by 0.3 points, VADv2 by 0.8 points, MeanFuser by 1.1 points, and GaussianFusion [49] by 1.3 points. It also obtains the best or tied-best scores on TTC, comfort, and EP, and ranks second on NC and DAC.

The heatmap-offset design provides spatial anchors for topology-aware coarse localization, while the continuous offset branch refines trajectory geometry. Hierarchical conditional cross-attention aligns scene semantics with navigation intent, and piecewise consistency regularizes long-horizon generation. Qualitative visualizations in Appendix H show compact multimodal candidates in intersection and turning scenarios.

Results on NAVSIMv2. Table 2 shows the results on NAVSIMv2 under one-stage, non-reactive evaluation. TwinFlux achieves the best EPDMS of 90.5, surpassing MeanFuser by 1.0 points and DiffusionDrive by 2.2 points. It leads NC, EP, TTC, lane keeping, and EC, ties the best DDC, traffic-light compliance, and heading comfort, and ranks second on DAC.

The NAVSIMv2 results indicate that TwinFlux remains effective under the extended metric suite. Directly predicting the absolute heading angle in the continuous offset branch avoids estimating heading from finite coordinate differences and is intended to support lane keeping and heading control. Dual-timestep affine modulation and piecewise trajectory consistency regularize temporal generation.

Table 3: Model size, NAVSIMv1 performance, and RTX 3090 speed.

Method	Params (M)	PDMS ↑	FPS ↑
TransFuser [5]	55.9	84.0	**41**
GoalFlow [28]	62.3	85.7	9
Hydra-MDP [9]	65.2	86.5	12
DiffusionDrive [10]	60.7	88.1	27
GaussianFusion [49]	51.6	88.8	30
MeanFuser [31]	54.6	<u>89.0</u>	35
TwinFlux	58.1	**90.1**	<u>37</u>

Table 4: Core module and representation ablation. Best values are highlighted within each trajectory-paradigm block.

Paradigm	Rep.	HCC	DAM	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
MLP	Coord.	–	–	**97.7**	**92.8**	**93.8**	**99.8**	**79.6**	**84.0**
	H+O	–	–	97.3	90.9	91.4	**99.8**	78.0	82.6
Diffusion	Coord.	–	–	**98.2**	**96.2**	**94.9**	**100.0**	**82.1**	**88.1**
	H+O	–	–	**98.2**	96.0	94.6	**100.0**	79.9	87.0
SplitMeanFlow	Coord.	–	–	98.5	97.2	96.2	**100.0**	82.6	88.8
	H only	–	–	97.3	95.4	95.1	**100.0**	78.8	86.7
	O only	–	–	98.6	97.1	95.9	**100.0**	82.7	88.9
	H+O	–	–	98.7	<u>97.3</u>	96.4	**100.0**	82.5	89.3
	H+O	✓	–	<u>98.9</u>	<u>97.3</u>	<u>96.5</u>	**100.0**	<u>83.1</u>	<u>89.8</u>
	H+O	✓	✓	**99.1**	**97.5**	**96.9**	**100.0**	**83.7**	**90.1**

Inference efficiency. Table 3 compares parameter size, NAVSIMv1 planning performance, and end-to-end inference speed on a single NVIDIA RTX 3090. TwinFlux achieves the highest PDMS with near-best FPS and a parameter size comparable to existing ResNet-34 planners. One-step average-flow generation and parallel heatmap-offset branches avoid multi-step denoising and serial refinement during inference. At batch size 1, planning takes 2.98 ms and the full camera–LiDAR system takes 27.38 ms, averaged over 1000 passes (Appendix F).

Statistical reliability. Five independently trained NAVSIMv1 runs yield 90.11 ± 0.06 PDMS (mean and sample SD). Paired scene-bootstrap 95% CIs for the gains over reproduced MeanFuser are $[0.49, 1.85]$ on NAVSIMv1 and $[0.37, 1.61]$ on NAVSIMv2 (Appendix G.5).

Reactive closed-loop evaluation. On the full official Bench2Drive evaluation split (devkit v0.0.3, CARLA 0.9.15), TwinFlux achieves 81.3 driving score, 56.1 success rate, and 58.4 mean ability score after benchmark-specific training. It leads emergency braking, giving way, and traffic-sign handling, and ranks second on merging and overtaking among the reported methods. The full table and paper-reported baseline provenance are in Appendix F.

4.4 Ablation Studies

We conduct ablations on NAVSIMv1 to evaluate the contribution of key TwinFlux components and the prototype source design. All variants use the same ResNet-34 backbone, training settings, and evaluation protocol.

Core module ablation. Table 4 studies the interaction between trajectory representation and prediction paradigm. Coord. denotes coordinate-space generation; H+O denotes full TDR; H only removes the offset branch; and O only fixes the prototype heatmap while predicting offsets. Directly replacing coordinates with H+O does not help MLP or diffusion baselines, whereas under SplitMeanFlow it improves PDMS from 88.8 to 89.3. Single-branch variants are weaker, and adding HCC and DAM further raises PDMS to 90.1 with the best NC, DAC, TTC, and EP, showing that TDR, semantic conditioning, and timestep modulation are complementary under one-step average-flow

Table 5: Matched representation–objective study and velocity-boundary controls on NAVSIMv1. Full metrics and protocols are in Appendices G.4 and G.2.

Representation	**Params (M)**	**Direct**	**SplitMeanFlow**
Coordinate residual	58.0	**86.9**	<u>88.7</u>
Heatmap + offset	58.1	<u>86.4</u>	**90.1**
Training setting	p_{vb}	**Interval consistency**	**PDMS**
Full	0.7	Yes	**90.1**
VB only	1.0	No	<u>88.8</u>
Drop interval loss	0.7	No	87.9

generation. Appendix G provides further ablations, including component-number and velocity-boundary analyses in Appendices G.1 and G.2.

Matched controls. With prototypes, modules, auxiliary losses, scoring labels, and training budget matched, heatmap–offset improves over coordinate residuals by 1.4 PDMS under SplitMeanFlow, but reduces it by 0.5 under direct endpoint supervision (Table 5). The interaction is $+1.9$ with paired 95% CI $[1.07, 2.81]$. Full training improves over VB-only by 1.30 (95% CI $[1.06, 1.53]$); removing interval consistency while retaining $p_{\mathrm{vb}} = 0.7$ gives 87.9. These controls support complementary representation and training effects.

Effect of the prototype source. Prototype-source ablations in Appendix G.3 show that fixed K-means prototypes outperform constant-velocity and stochastic source variants, supporting stable prototype-conditioned refinement from data-driven trajectory modes. Raw-prototype and far-prototype controls in Appendix G.6 distinguish source quality from scene-conditioned refinement; scorer controls and Oracle@K results are in Appendix G.7.

5 Conclusion

We propose TwinFlux, a discrete-continuous parallel average-flow framework for end-to-end autonomous driving trajectory planning. TwinFlux uses data-driven trajectory prototypes as source endpoints, decouples generation into topology-aware heatmaps and grid-relative offsets, and trains one-step prediction with SplitMeanFlow-based piecewise consistency in reconstructed trajectory space. Together with hierarchical conditional cross-attention, dual-timestep affine modulation, and trajectory scoring, TwinFlux achieves strong NAVSIMv1 and NAVSIMv2 planning performance while maintaining real-time inference.

Limitations. Our experiments cover NAVSIM and Bench2Drive with benchmark-specific training. Prototype coverage, candidate ranking, and generalization under domain, sensor, and map shifts, including rare interactions, remain directions for further study. Appendix I discusses objective-level limitations, broader impacts, and deployment considerations.

Acknowledgements and Disclosures

This work was supported in part by the National Natural Science Foundation of China under Grant 62476088 and the Science and Technology Commission of Shanghai Municipality (20DZ2254400). The authors declare no competing interests.

References

[1] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *Advances in Neural Information Processing Systems*, 35:6119–6132, 2022.

[2] Hao Shao, Letian Wang, Ruobing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Conference on Robot Learning*, pages 726–737. PMLR, 2023.

[3] Katrin Renz, Kashyap Chitta, Otniel-Bogdan Mercea, A Koepke, Zeynep Akata, and Andreas Geiger. Plant: Explainable planning transformers via object-level representations. *arXiv preprint arXiv:2210.14222*, 2022.

[4] Wenzhao Zheng, Ruiqi Song, Xianda Guo, Chenming Zhang, and Long Chen. Genad: Generative end-to-end autonomous driving. In *European Conference on Computer Vision*, pages 87–104. Springer, 2024.

[5] 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.

[6] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17853–17862, 2023.

[7] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8350, 2023.

[8] Bo Jiang, Shaoyu Chen, Hao Gao, Bencheng Liao, Qian Zhang, Wenyu Liu, and Xinggang Wang. VADv2: End-to-end vectorized autonomous driving via probabilistic planning. In *International Conference on Learning Representations*, 2026. URL `https://arxiv.org/abs/2402.13243`.

[9] 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.

[10] 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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12037–12047, 2025.

[11] Zhiyu Zheng, Shaoyu Chen, Haoran Yin, Xinbang Zhang, Jialv Zou, Xinggang Wang, Qian Zhang, and Lefei Zhang. Resad: Normalized residual trajectory modeling for end-to-end autonomous driving. *arXiv preprint arXiv:2510.08562*, 2025.

[12] Yinan Zheng, Ruiming Liang, Kexin Zheng, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, et al. Diffusion-based planning for autonomous driving with flexible guidance. *arXiv preprint arXiv:2501.15564*, 2025.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[14] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[15] 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.

[16] Yi Guo, Wei Wang, Zhihang Yuan, Rong Cao, Kuan Chen, Zhengyang Chen, Yuanyuan Huo, Yang Zhang, Yuping Wang, Shouda Liu, et al. Splitmeanflow: Interval splitting consistency in few-step generative modeling. *arXiv preprint arXiv:2507.16884*, 2025.

[17] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *European Conference on Computer Vision*, pages 533–549. Springer, 2022.

[18] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15793–15803, 2021.

[19] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021.

[20] ByeoungDo Kim, Seong Hyeon Park, Seokhwan Lee, Elbek Khoshimjonov, Dongsuk Kum, Junsoo Kim, Jeong Soo Kim, and Jun Won Choi. Lapred: Lane-aware prediction of multi-modal future trajectories of dynamic agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14636–14645, 2021.

[21] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019.

[22] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on robot learning*, pages 895–904. PMLR, 2021.

[23] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15303–15312, 2021.

[24] Ellington Kirby, Alexandre Boulch, Yihong Xu, Yuan Yin, Gilles Puy, Éloi Zablocki, Andrei Bursuc, Spyros Gidaris, Renaud Marlet, Florent Bartoccioni, Anh-Quan Cao, Nermin Samet, Tuan-Hung Vu, and Matthieu Cord. Driving on registers. *arXiv preprint arXiv:2601.05083*, 2026. URL `https://arxiv.org/abs/2601.05083`.

[25] Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Conditional flow matching: Simulation-free dynamic optimal transport. *arXiv preprint arXiv:2302.00482*, 2(3), 2023.

[26] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[27] Qi Yan, Brian Zhang, Yutong Zhang, Daniel Yang, Joshua White, Di Chen, Jiachao Liu, Langechuan Liu, Binnan Zhuang, Shaoshuai Shi, et al. Trajflow: Multi-modal motion prediction via flow matching. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3923–3928. IEEE, 2025.

[28] Zebin Xing, Xingyu Zhang, Yang Hu, Bo Jiang, Tong He, Qian Zhang, Xiaoxiao Long, and Wei Yin. Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1602–1611, 2025.

[29] Yuxiang Fu, Qi Yan, Lele Wang, Ke Li, and Renjie Liao. Moflow: One-step flow matching for human trajectory forecasting via implicit maximum likelihood estimation based distillation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17282–17293, 2025.

[30] Leandro Di Bella, Adrian Munteanu, and Bruno Cornelis. Flows: One-step motion prediction via local transport conditioning. *arXiv preprint arXiv:2604.26065*, 2026.

[31] Junli Wang, Yinan Zheng, Xueyi Liu, Zebin Xing, Pengfei Li, Guang Li, Kun Ma, Guang Chen, Hangjun Ye, Zhongpu Xia, et al. Meanfuser: Fast one-step multi-modal trajectory generation and adaptive reconstruction via meanflow for end-to-end autonomous driving. *arXiv preprint arXiv:2602.20060*, 2026.

[32] Lingguang Wang, Ömer Şahin Taş, Marlon Steiner, and Christoph Stiller. Flowdrive: moderated flow matching with data balancing for trajectory planning. *arXiv preprint arXiv:2509.21961*, 2025.

[33] Tianyi Tan, Yinan Zheng, Ruiming Liang, Zexu Wang, Kexin Zheng, Jinliang Zheng, Jianxiong Li, Xianyuan Zhan, and Jingjing Liu. Flow matching-based autonomous driving planning with advanced interactive behavior modeling. *arXiv preprint arXiv:2510.11083*, 2025.

[34] Yifang Xu, Jiahao Cui, Feipeng Cai, Zhihao Zhu, Hanlin Shang, Shan Luan, Mingwang Xu, Neng Zhang, Yaoyi Li, Jia Cai, and Siyu Zhu. WAM-Flow: Parallel coarse-to-fine motion planning via discrete flow matching for autonomous driving. *arXiv preprint arXiv:2512.06112*, 2025. URL `https://arxiv.org/abs/2512.06112`.

[35] Qiao Sun, Xin Huang, Junru Gu, Brian C Williams, and Hang Zhao. M2i: From factored marginal trajectory prediction to interactive prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6543–6552, 2022.

[36] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11553–11562, 2020.

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

[38] 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.

[39] Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-simulation for autonomous driving. In *Conference on Robot Learning*, 2025. URL `https://arxiv.org/abs/2506.04218`.

[40] Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2Drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL `https://arxiv.org/abs/2406.03877`.

[41] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[42] Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15449–15458, 2024.

[43] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 28632–28642, 2025.

[44] Kaiwen Zhang, Zhenyu Tang, Xiaotao Hu, Xingang Pan, Xiaoyang Guo, Yuan Liu, Jingwei Huang, Li Yuan, Qian Zhang, Xiao-Xiao Long, et al. Epona: Autoregressive diffusion world model for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27220–27230, 2025.

[45] 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.

[46] Chengran Yuan, Zhanqi Zhang, Jiawei Sun, Shuo Sun, Zefan Huang, Christina Dao Wen Lee, Dongen Li, Yuhang Han, Anthony Wong, Keng Peng Tee, et al. Drama: An efficient end-to-end motion planner for autonomous driving with mamba. *arXiv preprint arXiv:2408.03601*, 2024.

[47] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27137–27146, 2025.

[48] Zhenxin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Zuxuan Wu, and Jose M Alvarez. Hydra-next: Robust closed-loop driving with open-loop training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27305–27314, 2025.

[49] Shuai Liu, Quanmin Liang, Zefeng Li, Boyang Li, and Kai Huang. Gaussianfusion: Gaussian-based multi-sensor fusion for end-to-end autonomous driving. *arXiv preprint arXiv:2506.00034*, 2025.

[50] Rui Zhao, Yuze Fan, Ziguo Chen, Fei Gao, and Zhenhai Gao. Diffe2e: Rethinking end-to-end driving with a hybrid diffusion-regression-classification policy. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[51] 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.

[52] 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.

[53] Maciej Wozniak, Lianhang Liu, Yixi Cai, and Patric Jensfelt. Prix: learning to plan from raw pixels for end-to-end autonomous driving. *IEEE Robotics and Automation Letters*, 11(5): 6400–6407, 2026.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[55] 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. URL `https://arxiv.org/abs/2305.10430`.

[56] Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21983–21994, 2023.

[57] Xiaosong Jia, Junqi You, Zhiyuan Zhang, and Junchi Yan. DriveTransformer: Unified transformer for scalable end-to-end autonomous driving. In *International Conference on Learning Representations*, 2025. URL `https://proceedings.iclr.cc/paper_files/paper/2025/hash/a7afc9957f1190223763b6ea93218f98-Abstract-Conference.html`.

[58] Xiaosong Jia, Yulu Gao, Li Chen, Junchi Yan, Patrick Langechuan Liu, and Hongyang Li. DriveAdapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7953–7963, 2023.

[59] Julian Zimmerlin, Jens Beißwenger, Bernhard Jaeger, Andreas Geiger, and Kashyap Chitta. Hidden biases of end-to-end driving datasets. *arXiv preprint arXiv:2412.09602*, 2024. URL `https://arxiv.org/abs/2412.09602`.

A Additional Details of SplitMeanFlow

The main text introduces the SplitMeanFlow displacement-consistency objective used by TwinFlux. Here we only specify its instantiation for the heatmap-offset trajectory representation. For each training sample, the selected cluster prototype is the source endpoint and the expert trajectory is the data endpoint. After trajectory decoupling, $(H^*, O^*) = f_{\text{decouple}}(\tau^*)$, the bridge states at any flow time s are

$$Z_{h,s}^{k^*} = (1-s)Z_h^{k^*} + sH^*, \qquad Z_{o,s}^{k^*} = (1-s)Z_o^{k^*} + sO^*. \tag{20}$$

TwinFlux randomly samples $[t, r]$ with $t < r$, applies displacement consistency to the heatmap and offset latents over this interval, and aligns the resulting direct and accumulated predictions after trajectory reconstruction:

$$\hat{\tau}_{\text{direct}} = f_{\text{recon}}(\hat{H}_{\text{direct}}, \hat{O}_{\text{direct}}), \qquad \hat{\tau}_{\text{accum}} = f_{\text{recon}}(\hat{H}_{\text{accum}}, \hat{O}_{\text{accum}}). \tag{21}$$

In the corresponding loss, $\hat{\tau}_{\text{accum}}$ is detached with $\text{sg}(\cdot)$ and used as the target for $\hat{\tau}_{\text{direct}}$. For velocity-boundary samples, TwinFlux sets the interval length to zero and supervises $u_\theta(\cdot, F_c, q, 0)$ as an instantaneous velocity estimate of the same linear bridge at a sampled flow time q. All branch architectures and reconstruction details are defined in the main method and the following appendix sections.

B Implementation Details of Scene Encoding

We use the camera and rasterized BEV LiDAR inputs described in the main experiments section. The image and LiDAR streams are encoded in the same ego-centered BEV frame, projected to a shared channel dimension, and fused into

$$F_e = \text{Fuse}\big(F_{\text{img_bev}}, F_{\text{lidar_bev}}\big), \qquad F_e \in \mathbb{R}^{C \times H_b \times W_b}. \tag{22}$$

The ego state and high-level navigation command are encoded by a lightweight MLP,

$$F_s = \text{MLP}(s), \tag{23}$$

and the flow start time and interval are encoded separately as $F_t^1 = \text{PosEnc}(t)$ and $F_t^2 = \text{PosEnc}(\Delta t)$. TwinFlux uses compact scene tokens derived from F_e as the spatial condition in HCC, F_s as the ego-intent condition, and (F_t^1, F_t^2) as the temporal condition in DAM.

C Additional Details of the Discrete-Continuous Generator

For the discrete heatmap branch, the input source tensor $Z_h \in \mathbb{R}^{H_g \times W_g \times T}$ is encoded into one temporal generation token per future step:

$$X_h = \text{EncTok}_h(Z_h) \in \mathbb{R}^{T \times C}. \tag{24}$$

Here EncTok_h summarizes each $H_g \times W_g$ heatmap slice into a C-dimensional token while preserving the temporal ordering. Multi-head self-attention is therefore computed over the T future-step tokens:

$$Q_h = X_h W_q^h, \qquad K_h = X_h W_k^h, \qquad V_h = X_h W_v^h, \tag{25}$$

$$F_z^h = \text{SelfAttn}(X_h) = \text{Softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_k}}\right) V_h. \tag{26}$$

The continuous offset branch applies the same temporal tokenization to $Z_o \in \mathbb{R}^{H_g \times W_g \times T \times 3}$,

$$X_o = \text{EncTok}_o(Z_o) \in \mathbb{R}^{T \times C}, \tag{27}$$

and obtains F_z^o after self-attention.

In HCC, the query tokens come from the temporal generation sequence, whereas the environmental key/value tokens are compact scene tokens. Specifically, the dense BEV feature map $F_e \in \mathbb{R}^{C \times H_b \times W_b}$ is summarized before cross-attention as

$$S_e = \text{SceneTok}(F_e) \in \mathbb{R}^{N \times C}, \qquad N \ll H_b W_b. \tag{28}$$

The discrete heatmap branch first injects environmental features:

$$Q_{h,e} = F_z^h, \qquad K_{h,e} = S_e W_k^{h,e}, \qquad V_{h,e} = S_e W_v^{h,e}, \tag{29}$$

$$F_{z,e}^h = \mathrm{Softmax}\left(\frac{Q_{h,e} K_{h,e}^\top}{\sqrt{d_k}}\right) V_{h,e}. \tag{30}$$

The feature $F_{z,e}^h$ is then used as the query in a second cross-attention layer, where the expanded ego-state feature derived from F_s provides the keys and values. The output is $F_{z,e,s}^h$. The continuous offset branch follows the same design.

For Dual-Timestep Affine Modulation, the start time and time interval are encoded as

$$F_t^1 = \mathrm{PosEnc}(t), \qquad F_t^2 = \mathrm{PosEnc}(\Delta t). \tag{31}$$

Two MLP pairs generate affine parameters:

$$\gamma_t = \mathrm{MLP}_t^\gamma(F_t^1), \qquad \beta_t = \mathrm{MLP}_t^\beta(F_t^1), \tag{32}$$

$$\gamma_{\Delta t} = \mathrm{MLP}_{\Delta t}^\gamma(F_t^2), \qquad \beta_{\Delta t} = \mathrm{MLP}_{\Delta t}^\beta(F_t^2). \tag{33}$$

The modulated feature is computed by

$$F_{t1}^h = \gamma_t \odot F_{z,e,s}^h + \beta_t, \tag{34}$$

$$F_{t2}^h = \gamma_{\Delta t} \odot F_{t1}^h + \beta_{\Delta t}. \tag{35}$$

The same modulation is applied to the continuous offset branch to obtain F_{t2}^o.

Finally, two prediction heads decode the modulated temporal features back to the grid-structured velocity fields:

$$u_\theta^h(Z_h, F_c, t, \Delta t) = \mathrm{Head}_h(F_{t2}^h), \tag{36}$$

$$u_\theta^o(Z_o, F_c, t, \Delta t) = \mathrm{Head}_o(F_{t2}^o). \tag{37}$$

The heatmap head outputs $u_\theta^h \in \mathbb{R}^{H_g \times W_g \times T}$, and the offset head outputs $u_\theta^o \in \mathbb{R}^{H_g \times W_g \times T \times 3}$. A separate score head predicts the confidence of the candidate trajectory from both modulated branch features:

$$\ell_\rho = \mathrm{ScoreHead}\big(\mathrm{Pool}\big([F_{t2}^h; F_{t2}^o]\big)\big), \qquad \hat{\rho} = \mathrm{Sigmoid}(\ell_\rho). \tag{38}$$

This score head is applied independently to each prototype-conditioned candidate. The temporal tokenization makes self-attention scale with T tokens and environmental cross-attention scale with N compact scene tokens, rather than with all $H_g W_g T$ generation-grid cells and all $H_b W_b$ BEV cells. The special case $\Delta t = 0$ is used only for velocity-boundary supervision.

The best-mode assignment in the main method provides binary labels $y_k = \mathbb{I}(k = k^*)$ for all prototype-conditioned candidates. The score head is trained with binary cross-entropy:

$$\mathcal{L}_{\mathrm{score}} = -\frac{1}{K} \sum_{k=1}^{K} \left[y_k \log \hat{\rho}_k + (1 - y_k) \log(1 - \hat{\rho}_k) \right]. \tag{39}$$

D Reversibility of Trajectory Decoupling and Reconstruction

We provide a short verification of the reversibility of TDR. All grid anchors in TDR are defined on the trajectory generation grid (H_g, W_g) with physical resolution Δs_g and origin (u_0^g, v_0^g). The physical coordinate of grid cell (u, v) is

$$g_x(u, v) = (v - v_0^g)\Delta s_g, \qquad g_y(u, v) = (u_0^g - u)\Delta s_g. \tag{40}$$

For a trajectory state $\tau_i = [x_i, y_i, \theta_i]$, the decoupling module defines the global exact residuals

$$\delta x_i(u, v) = x_i - g_x(u, v), \qquad \delta y_i(u, v) = y_i - g_y(u, v), \qquad \delta \theta_i(u, v) = \theta_i. \tag{41}$$

During reconstruction, the gridwise coordinate predictions are

$$x_i^{\mathrm{pred}}(u, v) = g_x(u, v) + \delta x_i(u, v), \tag{42}$$

$$y_i^{\text{pred}}(u,v) = g_y(u,v) + \delta y_i(u,v). \tag{43}$$

Substituting the residual definitions gives

$$x_i^{\text{pred}}(u,v) = g_x(u,v) + x_i - g_x(u,v) = x_i, \tag{44}$$

$$y_i^{\text{pred}}(u,v) = g_y(u,v) + y_i - g_y(u,v) = y_i. \tag{45}$$

Therefore, for any normalized probability map P_i satisfying $\sum_u \sum_v P_i(u,v) = 1$,

$$\hat{x}_i = \sum_{u=1}^{H_g} \sum_{v=1}^{W_g} P_i(u,v) x_i = x_i, \tag{46}$$

$$\hat{y}_i = \sum_{u=1}^{H_g} \sum_{v=1}^{W_g} P_i(u,v) y_i = y_i, \tag{47}$$

and

$$\hat{\theta}_i = \sum_{u=1}^{H_g} \sum_{v=1}^{W_g} P_i(u,v) \theta_i = \theta_i. \tag{48}$$

Thus, $\hat{\tau}_i = \tau_i$ and $f_{\text{recon}}(f_{\text{decouple}}(\tau)) = \tau$. This property relies on the global exact residual definition; local residual assignment would not guarantee the same lossless reconstruction under soft aggregation.

E Auxiliary Perception Losses

The auxiliary perception decoder predicts surrounding-object detections and lane maps from the environmental feature F_e. Let (H_p, W_p) denote the output resolution of the auxiliary perception heads.

The detection loss combines focal loss for classification and Smooth L_1 loss for bounding-box regression:

$$\mathcal{L}_{\text{det}} = \alpha_{\text{fl}} \text{FocalLoss}(\hat{y}, y) + \alpha_{\text{reg}} \text{SmoothL1}(\hat{b}, b). \tag{49}$$

The lane-map loss uses pixelwise cross-entropy:

$$\mathcal{L}_{\text{map}} = -\frac{1}{H_p W_p} \sum_{i=1}^{H_p} \sum_{j=1}^{W_p} \sum_{c=1}^{C_{\text{map}}} y_{i,j,c} \log \hat{y}_{i,j,c}. \tag{50}$$

The total auxiliary perception loss is

$$\mathcal{L}_{\text{perc}} = \lambda_1 \mathcal{L}_{\text{det}} + \lambda_2 \mathcal{L}_{\text{map}}. \tag{51}$$

F Experimental Protocol and Implementation Details

NAVSIM experiments follow the official NAVSIMv1 and NAVSIMv2 protocols [38], using the public navtest splits, v1.1/v2.2 devkits, and v1/v2 metric caches, respectively. Traffic is non-reactive in both; NAVSIMv2 uses one-stage evaluation. The main text reports the default NAVSIM input resolutions, training hardware, and inference setting; here we provide metric formulas, implementation details, and the additional Bench2Drive protocol.

Comparison scope and provenance. The NAVSIM main tables represent the public-method snapshot available by April 24, 2026. Baselines are paper- or leaderboard-reported unless explicitly identified as reproduced; MeanFuser is reproduced for paired scene-level comparisons in Appendix G.5. For example, WAM-Flow [34] reports 89.1/90.3 NAVSIMv1 PDMS with one/five steps using a 1.5B VLA backbone, metric-aligned discrete tokens, discrete flow matching, and simulator-guided GRPO. TwinFlux is a 58.1M camera–LiDAR planner using heatmap–offset SplitMeanFlow without JVPs, reaching 90.1 in one step.

Table 6: Ablation of the number of multimodal components.

K	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
1	96.5	95.1	92.7	99.8	80.6	87.1
2	97.2	96.2	93.4	99.8	81.7	87.9
6	<u>98.7</u>	97.3	<u>95.1</u>	**100.0**	82.9	89.3
10	**99.1**	**97.5**	**96.9**	**100.0**	**83.7**	**90.1**
14	98.4	<u>97.4</u>	94.8	**100.0**	<u>83.2</u>	<u>89.5</u>
20	98.1	96.9	94.6	<u>99.9</u>	83.0	89.0

Evaluation metrics. NAVSIMv1 uses PDMS:

$$\mathrm{PDMS}_i = NC_i \cdot DAC_i \cdot \frac{5EP_i + 5TTC_i + 2\mathrm{Comf}_i}{12}. \tag{52}$$

NAVSIMv2 uses EPDMS:

$$\mathrm{EPDMS}_i = NC_i \cdot DAC_i \cdot DDC_i \cdot TLC_i \cdot \frac{5EP_i + 5TTC_i + 2LK_i + 2HC_i + 2EC_i}{16}. \tag{53}$$

Here NC, DAC, DDC, and TLC are multiplicative safety or rule-compliance terms, while EP, TTC, Comf, LK, HC, and EC measure progress, time-to-collision margin, and comfort or lane-keeping quality. Final benchmark scores are computed by the official evaluation scripts.

Resolution notation. We distinguish the BEV feature grid (H_b, W_b) from the trajectory generation grid (H_g, W_g) used by the heatmap branch, offset branch, and TDR. The generation grid has physical resolution Δs_g and ego-centered origin (u_0^g, v_0^g). In the default setting, $H_b = W_b = 256$, $H_g = W_g = 32$, and the planning horizon is $T = 8$ steps over 4 seconds. When the two grids differ, HCC attends over T temporal generation tokens and N compact scene tokens rather than all cells from either grid.

Optimizer and learning-rate schedule. NAVSIM training uses the AdamW optimizer [41] with a weight-decay coefficient of 0.1. The initial learning rate is 2×10^{-4} and the minimum learning rate is 1×10^{-6}. We use a cosine annealing schedule with a three-epoch linear warm-up, during which the learning rate increases from 0 to the initial value and then decays smoothly following the cosine schedule.

Network architecture. The model embedding dimension is set to 128. The environment encoder processes the concatenated camera image with a ResNet-34 backbone [54], projects image features into BEV space using deformable attention with predefined BEV queries, and fuses them elementwise with rasterized BEV LiDAR features. The ego-state encoder is a two-layer MLP. The timestep encoder uses sinusoidal positional encodings, and all conditioning features are projected to the same 128-dimensional embedding space.

Both TwinFlux generation branches use six-layer Transformer decoders with eight attention heads, model dimension 128, feed-forward hidden dimension 256, and a Pre-LN structure for stable optimization. The discrete heatmap and continuous offset branches share the same conditioning design but use different prediction heads.

Cluster prototypes and source construction. The default source endpoints are $K = 10$ K-means trajectory prototypes obtained from NAVSIM training demonstrations. No Gaussian jitter or stochastic intra-mode sampling is applied. The reversible trajectory decoupling function converts each prototype into heatmap-offset source fields:

$$(Z_h^k, Z_o^k) = f_{\mathrm{decouple}}(a^k). \tag{54}$$

This initializes the flow process from cluster-prototype endpoints rather than structureless white noise.

Training and inference configuration. NAVSIM training uses a per-GPU batch size of 32 on $8\times$ NVIDIA RTX 3090 GPUs for 100 epochs; a full run takes approximately 10 hours. Data augmentation includes random brightness and contrast changes and small random BEV rotations within $\pm 5^\circ$. During inference, the 10 prototype-conditioned candidates are generated in parallel and ranked by the score head.

Latency boundaries. On one RTX 3090 at batch size 1, latency is averaged over 1000 forward passes with augmentation and gradient computation disabled. Planning takes 2.98 ms (approximately 335 FPS), measured from encoded scene features through the generator, TDR, scorer, and top-1 trajectory selection. End-to-end latency is 27.38 ms (approximately 37 FPS), measured from camera–LiDAR input tensors through encoding and planning to top-1 output. Loading, I/O, and visualization are excluded. Planning accounts for $2.98/27.38 = 10.9\%$ of this total.

Reactive closed-loop protocol and results. We use Bench2Drive [40] devkit v0.0.3 with CARLA 0.9.15 and run the official evaluator on the full evaluation/test split. TwinFlux is trained for this benchmark, with prototypes recomputed once from its training distribution and then fixed. A full training cycle takes roughly one week. Table 7 gives driving score (DS), success rate (SR), and the five ability scores. Baseline results are paper-reported. The mean ability score is

$$\text{Mean} = \frac{\text{Merge} + \text{Overtake} + \text{EBrake} + \text{GiveWay} + \text{TSign}}{5}. \tag{55}$$

TwinFlux reaches 81.3 DS, 56.1 SR, and 58.4 Mean, leading EBrake/GiveWay/TSign and ranking second on Merge/Overtake among these results.

Table 7: Reactive closed-loop Bench2Drive results. Baselines are paper-reported. All metrics are higher-is-better; – denotes an unreported result.

Method	DS	SR	Merge	Overtake	EBrake	GiveWay	TSign	Mean
AD-MLP [55]	18.1	0.0	0.0	0.0	0.0	0.0	4.4	0.9
TCP [1]	40.7	15.0	16.2	20.0	20.0	10.0	7.0	14.6
VAD [7]	42.4	15.0	8.1	24.4	18.6	20.0	19.2	18.1
UniAD [6]	45.8	16.4	14.1	17.8	21.7	10.0	14.2	15.6
ThinkTwice [56]	62.4	33.2	27.4	18.4	35.8	50.0	54.4	37.2
DriveTransformer [57]	63.5	35.0	17.6	35.0	48.4	40.0	52.1	38.6
DriveAdapter [58]	64.2	33.1	28.8	26.4	48.8	50.0	56.4	42.1
VADv2 [8]	76.2	50.5	–	–	–	–	–	–
TF++ [59]	76.9	54.0	**48.8**	37.6	64.2	50.0	59.7	52.1
GaussianFusion [49]	<u>79.1</u>	<u>54.4</u>	36.6	**64.4**	<u>66.5</u>	<u>53.3</u>	<u>60.8</u>	<u>56.3</u>
TwinFlux	**81.3**	**56.1**	<u>41.9</u>	<u>63.5</u>	**68.2**	**55.6**	**62.9**	**58.4**

Existing assets and licenses. NAVSIM code and repository assets are Apache-2.0 unless otherwise specified[3]. Underlying nuPlan/OpenScene data retain their distribution terms, including CC BY-NC-SA 4.0 and the nuPlan Dataset License Agreement for Non-Commercial Use[4]. Bench2Drive's v0.0.3 repository specifies CC BY-NC-ND 4.0 unless otherwise stated[5]; CARLA 0.9.15 code uses the MIT license[6], with separately licensed assets retaining their own terms.

G Additional Ablation Details

G.1 Ablation of Multimodal Component Number

Table 6 reports the effect of varying the number of components K. A small K limits coverage of behaviors such as going straight, turning, following, and lane changes. Increasing K improves multimodal coverage, but excessive components can fragment the training signal because each frame provides only one expert trajectory. Empirically, $K = 10$ achieves the best balance.

G.2 Ablation of Velocity-Boundary Sampling Probability

Table 8 studies the sensitivity to the velocity-boundary sampling probability p_{vb}. This hyperparameter controls the fraction of training samples that are converted to zero-length intervals and supervised by

[3] https://github.com/autonomousvision/navsim

[4] https://github.com/OpenDriveLab/OpenScene

[5] https://github.com/Thinklab-SJTU/Bench2Drive/tree/0.0.3

[6] https://github.com/carla-simulator/carla/blob/0.9.15/LICENSE

Table 8: Ablation of the velocity-boundary sampling probability p_{vb}.

p_{vb}	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
0.1	88.1	71.7	76.6	99.8	58.9	60.2
0.3	97.5	96.1	93.1	99.8	82.0	87.9
0.5	98.9	97.4	96.3	**100.0**	**83.7**	89.8
0.7	99.1	**97.5**	**96.9**	**100.0**	**83.7**	**90.1**
0.9	**99.2**	**97.5**	96.8	**100.0**	83.4	90.0
1.0	98.3	97.0	96.1	**100.0**	83.5	88.8

Table 9: Ablation of prototype source construction with matched training and inference sources.

#	Source Center	Gaussian Sampling	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
1		None	**99.1**	**97.5**	**96.9**	**100.0**	**83.7**	**90.1**
2	K-means prototype	Cluster covariance Gaussian	98.9	97.4	95.9	**100.0**	83.1	89.7
3		Isotropic Gaussian jitter	98.3	96.8	95.0	**100.0**	82.8	88.6
4	Constant-velocity extrapolation	None	97.9	94.7	92.9	**100.0**	80.6	85.4

the instantaneous velocity-boundary loss. The limiting case $p_{\mathrm{vb}} = 0$ removes this endpoint-induced velocity supervision, reducing training to interval self-consistency and admitting self-consistent but incorrect solutions. A very small value, e.g., $p_{\mathrm{vb}} = 0.1$, is still insufficient and causes severe degradation. Performance improves once enough velocity-boundary samples are included and peaks at $p_{\mathrm{vb}} = 0.7$, while $p_{\mathrm{vb}} = 0.9$ remains competitive but slightly reduces progress and the final PDMS. The $p_{\mathrm{vb}} = 1.0$ control uses only velocity-boundary flow supervision and reaches 88.8 PDMS.

Table 10: Velocity-boundary and interval-consistency controls. Other losses and architecture remain fixed.

Setting	p_{vb}	NC	DAC	TTC	Comf.	EP	PDMS
Full	0.7	99.1	**97.5**	**96.9**	100.0	**83.7**	**90.1**
Drop interval loss	0.7	98.5	96.7	95.6	100.0	82.1	87.9
Reduced consistency	0.9	**99.2**	**97.5**	96.8	100.0	83.4	90.0
VB only	1.0	98.3	97.0	96.1	100.0	83.5	88.8

The drop control uses the same $p_{\mathrm{vb}} = 0.7$ sampling and velocity-boundary supervision; full training additionally applies interval consistency to the remaining 30% of samples, improving PDMS by 2.2. Against VB-only training ($p_{\mathrm{vb}} = 1.0$), full and reduced consistency give $+1.30$ (95% CI $[1.06, 1.53]$) and $+1.21$ ($[1.01, 1.42]$), respectively, using the paired bootstrap in Appendix G.5. These controls show the complementary roles of velocity-boundary supervision and interval consistency on the expert-anchored bridge. Accumulated predictions are evaluated at prescribed states on this bridge.

G.3 Discussion of Prototype Source Construction

Table 9 studies how source construction affects one-step generation when the same source type is used during training and inference. For K-means prototype sources, we use $K = 10$ by default; the sensitivity study over K is provided in Appendix G.1.

The prototype-source ablation shows that clustered prototypes are important for TwinFlux. With matched training and inference sources, fixed K-means prototypes outperform constant-velocity extrapolation, indicating that expert-trajectory clusters provide a better initialization manifold for one-step generation. They capture driving intents such as turning, lane changes, and obstacle avoidance, whereas constant-velocity extrapolation mainly reflects local kinematic inertia.

Stochastic variants around K-means prototypes aim to introduce intra-mode variation. However, under best-mode imitation, the selected mode is still paired with a single expert trajectory for each sample. Perturbations around the same prototype therefore do not receive distinct submode-level endpoints, and the consistency objective encourages them to map to the same expert endpoint. Cluster-covariance Gaussian sampling is more effective than isotropic Gaussian jitter because its variance is estimated from demonstrations within each mode, but both add source variance without separate

supervision for the induced submodes. This supports using fixed cluster prototypes as stable source endpoints for scene-conditioned refinement.

G.4 Matched Representation–Objective and Heatmap Controls

Table 11 isolates representation and objective with matched prototypes, conditioning modules, auxiliary losses, score labels, optimization, and training budget, and nearly matched parameter counts. Direct bridge-end prediction fixes $(t, \Delta t) = (0, 1)$ and minimizes $\|\hat{\tau}^{k^*} - \tau^*\|_2^2$ over the full reconstructed T-step expert trajectory, without interval or velocity-boundary supervision. The coordinate-residual SplitMeanFlow baseline here retains the matched modules; its 88.7 PDMS is distinct from the module-removed coordinate baseline (88.8) in Table 4.

Table 11: Matched 2×2 representation–objective controls on NAVSIMv1.

ID	Representation	Objective	M params	NC	DAC	TTC	Comf.	EP	PDMS
A	Coordinate residual	Direct	58.0	98.0	95.8	95.7	**100.0**	82.0	86.9
B	Coordinate residual	SplitMeanFlow	58.0	<u>98.3</u>	<u>96.9</u>	<u>96.1</u>	**100.0**	<u>82.5</u>	<u>88.7</u>
C	Heatmap + offset	Direct	58.1	97.4	95.5	95.6	<u>99.8</u>	82.0	86.4
D	Heatmap + offset	SplitMeanFlow	58.1	**99.1**	**97.5**	**96.9**	**100.0**	**83.7**	**90.1**

Writing the PDMS of row q as P_q, the representation effects are $P_C - P_A = -0.5$ under direct supervision and $P_D - P_B = +1.4$ under SplitMeanFlow; objective effects are $P_B - P_A = +1.8$ in coordinates and $P_D - P_C = +3.7$ with heatmap–offset. The interaction is

$$\mathcal{I} = (P_D - P_B) - (P_C - P_A) = +1.9. \tag{56}$$

Paired 95% CIs are $[0.92, 1.85]$ for $P_D - P_B$ and $[1.07, 2.81]$ for $\mathcal{I}$. For each of $10{,}000$ bootstrap replicates, the same sampled aligned scene indices are used across all four checkpoints; the interaction is computed per scene before averaging. These intervals describe scene-sampling uncertainty for the fixed checkpoints.

Functional heatmap dependence. If $\mathbf{g}_j + \hat{\mathbf{O}}_j = \tau + \epsilon_j$ at one future step (with zero heading component in $\mathbf{g}_j$), normalized reconstruction gives

$$\hat{\tau} - \tau = \sum_j \hat{P}_j \epsilon_j. \tag{57}$$

Exact offsets make reconstruction heatmap-invariant (Appendix D); with learned, imperfect offsets, heatmap weights can change the reconstruction error. Table 12 intervenes only on reconstruction heatmaps, holding offsets, scores, selected candidate index, checkpoint, scenes, and evaluator fixed. Cyclic shifts apply the same direction and distance to all future steps, averaged over four $\pm x / \pm y$ runs. Spatial shuffle independently permutes each future heatmap and reports mean and sample SD ($n - 1$ denominator) over ten inference perturbation seeds. Together, the matched study and heatmap interventions show positive TDR–SplitMeanFlow interaction and functional dependence on learned heatmap weights.

Table 12: Reconstruction heatmap interventions on NAVSIMv1. Shift distance d is in generation-grid cells.

Heatmap	NC	DAC	TTC	Comf.	EP	PDMS
Predicted	**99.1**	**97.5**	**96.9**	100.0	**83.7**	**90.1**
Shift $d = 1$	<u>99.0</u>	<u>97.3</u>	<u>96.7</u>	100.0	<u>83.6</u>	<u>89.9</u>
Shift $d = 2$	98.6	97.1	96.3	100.0	83.0	89.5
Shift $d = 4$	97.9	96.3	95.5	100.0	81.9	88.0
Uniform	97.7	96.1	95.3	100.0	81.3	87.9
Shuffle (10 seeds)	97.6 ± 0.1	96.1 ± 0.2	95.3 ± 0.2	100.0 ± 0.0	81.2 ± 0.3	87.9 ± 0.2

G.5 Training Variability and Scene-Level Uncertainty

Table 13 reports five independent NAVSIMv1 training runs with identical settings except the random seed. The main result uses Seed 1, rounded to one decimal place. The mean 90.11 ± 0.06 uses sample SD $s = \sqrt{\sum_{r=1}^{n}(m_r - \bar{m})^2/(n-1)}$ with $n = 5$. This measures training variability; the scene bootstrap below instead holds checkpoints fixed.

Table 13: Five independently trained TwinFlux runs on NAVSIMv1; the final row is mean $\pm$ sample SD.

Run	NC	DAC	TTC	Comf.	EP	PDMS
Seed 1	99.14	97.51	96.89	100.00	83.67	90.13
Seed 2	99.10	97.43	96.85	100.00	83.74	90.11
Seed 3	99.11	97.46	96.97	100.00	83.69	90.08
Seed 4	99.26	97.64	97.02	100.00	83.81	90.19
Seed 5	99.02	97.33	96.85	100.00	83.71	90.04
Mean $\pm$ SD	99.13 ± 0.09	97.47 ± 0.11	96.92 ± 0.08	100.00 ± 0.00	83.72 ± 0.05	90.11 ± 0.06

For marginal intervals, each of $10{,}000$ bootstrap replicates samples N scene indices with replacement and averages the sampled scores. For paired intervals, scene tokens are aligned across checkpoints and the same sampled indices are applied to both methods. For scene scores m_i, the paired estimate is

$$\widehat{\Delta} = \frac{1}{N}\sum_{i=1}^{N}\left(m_i^{\mathrm{TwinFlux}} - m_i^{\mathrm{baseline}}\right). \tag{58}$$

The 2.5th and 97.5th percentiles of replicate means form each 95% CI (Table 14). Both paired MeanFuser intervals exclude zero. DiffE2E is compared using its published aggregate. The same resampling procedure is used for the controlled comparisons and far-prototype slice, reusing indices across all relevant checkpoints.

Table 14: Fixed-checkpoint scene-bootstrap uncertainty. Marginal rows report scores; paired rows report score differences against reproduced MeanFuser.

Dataset	Score or comparison	Estimate	95% CI
NAVSIMv1	TwinFlux PDMS	90.1	$[89.98, 90.24]$
NAVSIMv1	TwinFlux $-$ MeanFuser	$+1.1$	$[0.49, 1.85]$
NAVSIMv2	TwinFlux EPDMS	90.5	$[90.44, 90.67]$
NAVSIMv2	TwinFlux $-$ MeanFuser	$+1.0$	$[0.37, 1.61]$

G.6 Raw Prototypes, Refinement, and Coverage Stress Test

Table 15 compares raw prototypes and refined trajectories at the same learned candidate index, isolating the effect of scene-conditioned refinement for that selection. It raises PDMS from 74.4 to 90.1 overall and from 62.3 to 85.9 on far-prototype scenes. Raw and refined Oracle@10 rows independently select the highest-PDMS trajectory from their respective ten-candidate sets per scene and report all metrics from that trajectory; these are separate candidate-set upper bounds.

The stress-test slice is fixed before examining model differences: select the top 25% by expert-to-nearest-prototype distance, using the same boundary and tie-breaking rule across comparisons. TwinFlux and reproduced MeanFuser use identical scene tokens. MeanFuser scores 84.3 on this slice versus TwinFlux's 85.9, a $+1.6$ difference with paired 95% CI $[0.81, 2.37]$ from $10{,}000$ resamples. Both decline from their full-set scores (89.0 and 90.1) by 4.7 and 4.2, respectively. This subset evaluates scene-conditioned refinement when expert trajectories lie farther from their nearest prototypes. For another benchmark, source prototypes are recomputed once from its training distribution and then remain fixed.

G.7 Candidate Coverage, Scorer Controls, and Calibration

Following the distinction between candidate generation and scoring emphasized by DrivoR [24], Table 16 separates candidate coverage, inference-time ranking, and score-loss training effects. Or-

Table 15: Raw source versus scene-conditioned refinement on NAVSIMv1. Learned-index pairs use the same candidate index. Far denotes the predefined far-prototype top-25% slice.

Scenes	Output	Selection	NC	DAC	TTC	Comf.	EP	PDMS
All	Raw prototype	Learned	97.0	83.7	89.3	99.6	70.5	74.4
All	Refinement	Learned	99.1	97.5	96.9	**100.0**	83.7	90.1
All	Raw prototype	Oracle@10	97.2	84.1	91.8	99.6	71.2	75.8
All	Refinement	Oracle@10	**99.5**	**98.4**	**98.0**	**100.0**	**88.3**	**92.6**
Far	Raw prototype	Learned	91.7	77.5	83.4	99.6	61.7	62.3
Far	Refinement	Learned	**97.3**	**94.9**	**93.9**	**100.0**	**78.4**	**85.9**

acle@K chooses the highest-PDMS trajectory within the learned top-K candidates per scene; all reported metrics come from that single selected trajectory. The fixed selector always uses the most frequent training cluster; the random selector uniformly samples from the same ten candidates per scene, with mean and sample SD over ten inference seeds. The no-score-loss control retrains the same architecture and other losses and evaluates the same fixed cluster. Thus $90.1 - 77.5 = 12.6$ measures selection, while $77.5 - 77.1 = 0.4$ measures the score loss's training effect under fixed selection.

Table 16: Scorer and candidate-set controls on NAVSIMv1.

Selection/training setting	NC	DAC	TTC	Comf.	EP	PDMS
Learned top-1	99.1	97.5	96.9	**100.0**	83.7	90.1
Oracle@2	99.2	97.8	97.5	**100.0**	85.6	91.5
Oracle@6	**99.5**	98.3	97.9	**100.0**	**88.3**	92.4
Oracle@10	**99.5**	**98.4**	**98.0**	**100.0**	**88.3**	**92.6**
Most-frequent cluster (fixed)	97.7	85.1	93.4	99.7	72.2	77.5
Uniform random (10 seeds)	97.4 ± 0.1	85.1 ± 0.1	93.0 ± 0.2	99.7 ± 0.1	72.0 ± 0.2	77.4 ± 0.2
No $\mathcal{L}_{\text{score}}$, same fixed cluster	97.3	85.1	92.9	99.7	72.0	77.1

For $K = 10$, mean pairwise trajectory and endpoint distances are 2.3 m and 5.2 m. They average all 45 unordered candidate pairs per scene and then scenes; trajectory distance additionally averages the eight future steps. Together with the Oracle@10 gain, these indicate distinct and useful alternatives.

For a task-tolerant calibration diagnostic, let $c_i = \max_k \hat{\rho}_{ik}$ be the maximum sigmoid confidence and define success by a selected-to-oracle PDMS gap of at most 5:

$$y_i = \mathbf{1}\big[\mathrm{PDMS}_i(k_i^{\mathrm{oracle}}) - \mathrm{PDMS}_i(k_i^{\mathrm{sel}}) \le 5\big], \qquad \mathrm{ECE} = \sum_{b=1}^{5} \frac{|I_b|}{N} \left|\mathrm{acc}(I_b) - \mathrm{conf}(I_b)\right|. \quad (59)$$

Using five equal-width confidence bins I_b gives ECE 0.11, success base rate 0.81, and mean confidence 0.91, indicating mild overconfidence. This diagnostic assesses confidence calibration for near-oracle trajectory selection.

H Additional Qualitative Analysis

Figure 4 presents qualitative comparisons on representative NAVSIMv1 scenarios. The red trajectory denotes the highest-scoring prediction, the green trajectory denotes the expert demonstration, and lighter trajectories represent other multimodal candidates. TwinFlux generates compact candidate sets near the expert behavior. The discrete heatmap branch provides spatial anchors, while the continuous offset branch refines geometry near lane boundaries and intersections.

Figures 5 and 6 show more challenging scenarios, including high-curvature turns and interactive road segments. These views illustrate predicted trajectory geometry within the visible road corridor.

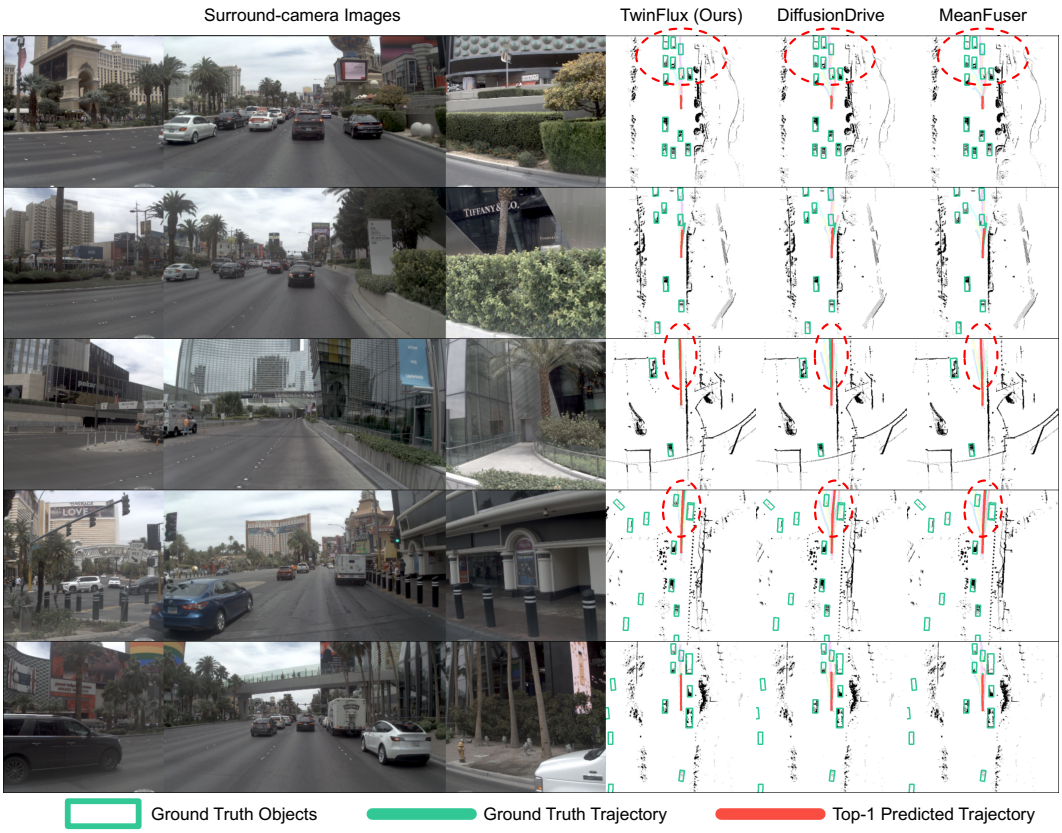


Figure 4: Qualitative comparison results on the NAVSIMv1 dataset.

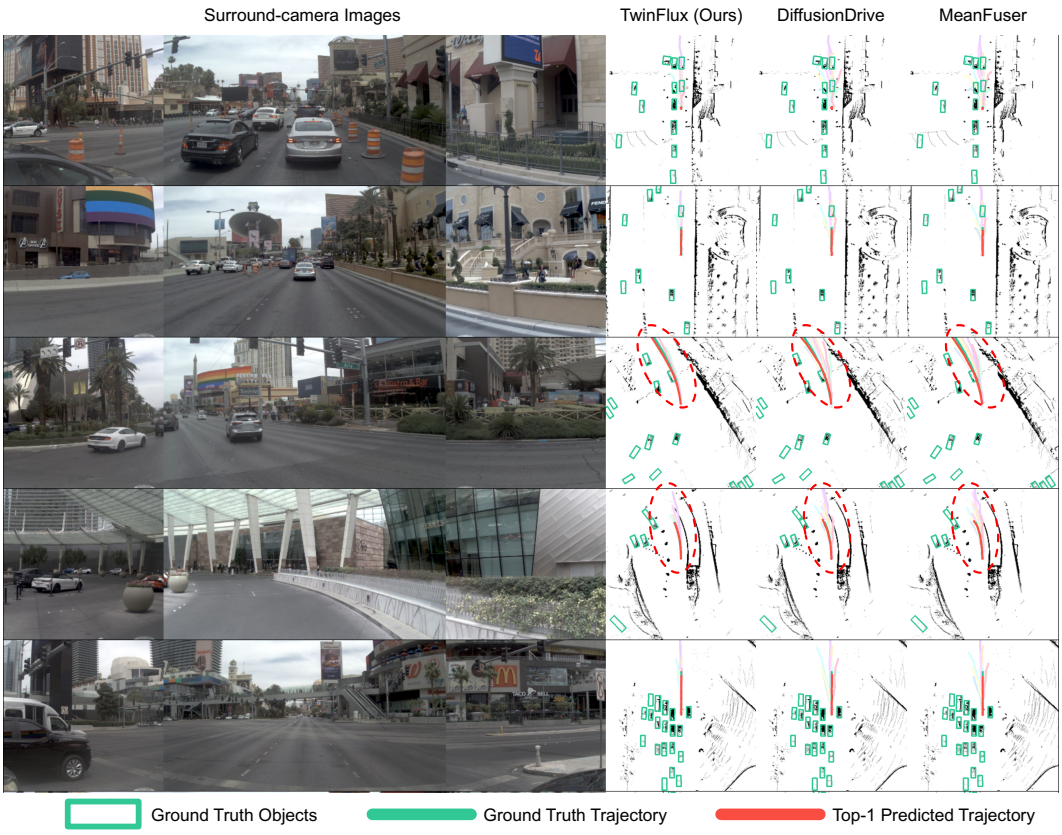


Figure 5: Qualitative comparison results of complex scenarios on the NAVSIMv1 dataset.



Figure 6: Additional qualitative comparison results of complex scenarios on the NAVSIMv1 dataset.

I Limitations and Broader Impacts

Evaluation and objective limits. Our evaluation combines non-reactive NAVSIM metrics with reactive closed-loop Bench2Drive simulation after benchmark-specific training. K-means prototypes summarize the training distribution; coverage of rare or unseen maneuvers remains a challenge. Flow consistency is evaluated on an expert-anchored bridge and depends on the source–target coupling and interpolation path. Generalization beyond this bridge and robustness to domain and weather shifts, sensor degradation, and map/lane convention mismatch remain directions for further study.

Failure modes and broader impacts. Efficient multimodal planning may lower inference cost and support useful driving assistance. Score mis-ranking and the observed overconfidence can select an unsuitable trajectory, creating safety risks for road users, especially in rare interactions. Deployment requires independent safety validation, runtime monitoring, fallback planning, and human oversight.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction summarize the proposed TwinFlux framework, its main components, and the NAVSIM evaluation scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of this work in the conclusion and Appendix I.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Appendix A provides the SplitMeanFlow consistency background used by TwinFlux, and Appendix D gives the reversibility analysis of trajectory decoupling and reconstruction.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper specifies the datasets, official protocols, architecture, training configuration, evaluation metrics, and implementation details in Section 4 and Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code, configurations, and model checkpoints will be made available at `https://github.com/ECUST-MMAI/TwinFlux`. Public benchmark access and evaluation protocols are described in Appendix F.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 4 and Appendix F report the input resolution, BEV and generation grids, planning horizon, optimizer, learning-rate schedule, batch size, epochs, hardware, and inference protocol.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Appendix G.5 reports five-training-seed mean and sample SD and defines marginal and paired scene-bootstrap 95% confidence intervals. Appendices G.4, G.2, and G.6 report controlled-comparison intervals; perturbation-seed SD is identified separately.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix F reports the use of 8 RTX 3090 GPUs for training, batch size, number of epochs, and single-RTX-3090 inference and latency measurement.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses public autonomous-driving benchmark data and does not involve private data collection, human-subject experiments, or activities that violate the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Appendix I discusses the potential benefits and risks of efficient multimodal driving assistance, together with validation, monitoring, fallback planning, and human oversight.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The work does not release high-risk generative models, scraped datasets, or systems intended for unrestricted deployment.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Appendix F cites the benchmarks and links their license terms, distinguishing NAVSIM's Apache-2.0 code, nuPlan/OpenScene data terms, Bench2Drive v0.0.3's CC BY-NC-ND 4.0 repository license, and CARLA's MIT code license and separately licensed assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new dataset, benchmark, or standalone asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The work does not involve human-subject research and therefore does not require IRB approval.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method, experiments, and scientific contributions do not involve LLMs as an important, original, or non-standard component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.