Supplementary Material for NAVSIM: Data-Driven Non-Reactive Autonomous Vehicle Simulation and Benchmarking

Daniel Dauner[1,2] **Marcel Hallgarten**[1,5] **Tianyu Li**[3] **Xinshuo Weng**[4] **Zhiyu Huang**[4,6] **Zetong Yang**[3] **Hongyang Li**[3] **Igor Gilitschenski**[7,8] **Boris Ivanovic**[4] **Marco Pavone**[4,9] **Andreas Geiger**[1,2] **Kashyap Chitta**[1,2]

[1]University of Tübingen [2]Tübingen AI Center [3]OpenDriveLab at Shanghai AI Lab
[4]NVIDIA Research [5]Robert Bosch GmbH [6]Nanyang Technological University
[7]University of Toronto [8]Vector Institute [9]Stanford University

Abstract

In this **supplementary document**, we first collect an overview of inconsistencies in existing planning benchmarks. Next, we provide details on the NAVSIM implementation and the dataset used in our study. Moreover, we include descriptions of the baselines and present supplementary results. Finally, we provide information on the 2024 NAVSIM challenge and discuss the broader impact of our work.

1 Inconsistencies in nuScenes Planning Benchmarks

Designing a consistent and standardized evaluation pipeline is crucial to ensure fair comparisons in benchmarks. However, due to the lack of a standardized evaluation setup on nuScenes [1] for end-to-end driving, there have been inconsistencies in prior work [27, 11, 12]. Specifically, although the same metrics, such as ADE or collision rates, have been reported, the underlying evaluation protocols[1] often differ. These differences include how metric results are averaged over timestamps, which objects are filtered during metric calculation, which frames are masked during evaluation, and what resolution of the BEV grid is used to calculate collisions. Such subtle differences can lead to inaccurate conclusions, which emphasizes the importance of our NAVSIM benchmark with principled and standardized evaluation on an independent test server, which is kept open to new submissions.

2 Implementation Details: NAVSIM

This section provides additional details regarding the NAVSIM framework and metrics used during evaluation. The code of NAVSIM, including baselines and evaluation, is available under the Apache-2.0 license [8] and open-sourced on GitHub: https://github.com/autonomousvision/navsim.

2.1 Task Description for Driving Agents

Keyframe-Based Evaluation. Simulating high-dimensional sensor streams for cameras or LiDAR can be cumbersome. Therefore, in NAVSIM, the trajectory planned by the driving agent in the initial frame of the scene is kept fixed for the entire simulation horizon. Unlike traditional closed-loop benchmarks that only use a small initial segment of the trajectory before replanning [6], NAVSIM uses the trajectory as a whole for evaluation. Therefore, agents need to anticipate the movement of

[1]For details on inconsistencies in nuScenes end-to-end driving evaluation, please refer to PARA-Drive [26].

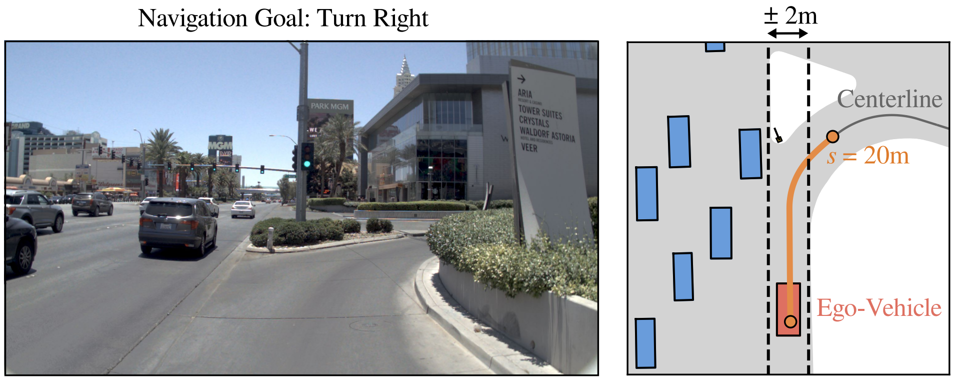

Figure 1: **Navigation Goal.** Given the map, we extract a route centerline (gray) to interpolate a goal point at a distance $s = 20$m (orange) from the rear axle of the ego-vehicle (red). The navigation goal is classified as a right turn, as the lateral distance exceeds 2m to the right.

surrounding agents and plan accordingly. We simulate the trajectory with a controller and motion model (see Sec. 2.2).

Sensor Input. Autonomous vehicles require sensor setups to capture details regarding their environment. At every time step, an agent in NAVSIM has access to 8 surround-view cameras, each with an image resolution of 1920×1080 pixels. Agents may additionally receive a point cloud merged from 5 LiDAR sensors mounted on the vehicle. While recent literature in open-loop planning has shifted away from using LiDAR [11, 12, 26], processing multi-view imagery during training and testing significantly increased compute requirements. We include LiDAR to enable exploration for more efficient architectures, utilizing fewer or lower-resolution camera inputs. Besides these sensors, agents have access to the ego velocity, acceleration, and navigation goal. This information is accessible for the current time step and 3 past frames, which are 0.5 seconds apart. Note that privileged information, such as the HD map, is only used in NAVSIM to unroll the simulation and compute metrics but is not accessible to the driving agents.

Navigation Goal. To disambiguate driver intention, we provide a discrete directional command as a one-hot vector (i.e., left, straight, right, and unknown). The command is based on the centerline in [6], which extracts the lane-center segments along the route using Dijkstra's algorithm on the map. We interpolate along the centerline for $s = 20$m from the ego position and classify the direction as left or right if the interpolated point exceeds a lateral threshold of ± 2m, as shown in Fig. 1. Otherwise, the direction is categorized as straight. Finally, it is set to unknown if the route is not available. The navigation goal in NAVSIM does not incorporate information about obstacles or human operator behavior, unlike previous benchmarks [10–12] based on nuScenes [1].

2.2 Predictive Driver Model Score (PDMS)

Similar to established closed-loop benchmarks [7, 13], NAVSIM evaluates the driving performance of agents with a single aggregated score called Predictive Driver Model Score (PDMS) $\in [0, 1]$. This scoring function closely resembles nuPlan's closed-loop score (CLS) [13] and originates from the PDM-Closed planner, the winner of the 2023 nuPlan competition. The non-reactive simulation and metrics for PDMS are highly optimized due to the strict run-time requirements of the nuPlan challenge. We use a frequency of 10Hz for the simulation and scoring pipeline. As OpenScene provides frames at 2Hz, we interpolate agent bounding boxes to a resolution of 10Hz. As observed in previous work [1, 20], we find that upsampling

Metric	**Weight**	**Range**
No at-fault Collision (NC)	-	$\{0, \frac{1}{2}, 1\}$
Drivable Area Compl. (DAC)	-	$\{0, 1\}$
Time to Collision (TTC)	5	$\{0, 1\}$
Ego Progress (EP)	5	$[0, 1]$
Comfort (C)	2	$\{0, 1\}$

Table 1: **PDMS.** Subscores with weights and ranges.

the temporal resolution is sufficient for scoring, and beneficial given the substantial reduction in storage requirements. In the following, we provide further details on the simulation and the subscores, summarized in Table 1.

Non-Reactive Simulation. The simulation ensures that a driving agent is evaluated on vehicle movement that is kinematically feasible. Given that the trajectory of a driving agent provides poses with time steps, we first interpolate the trajectory to the simulation frequency of 10Hz. Over the simulation horizon of $h = 4$s, we execute a Linear Quadratic Regulator (LQR) [15] controller to calculate acceleration and steering values and propagate the ego pose with a kinematic bicycle model [21]. We use the same controller parameters of PDM-Closed and nuPlan [6, 13].

No at-fault Collisions (NC). The background agents in NAVSIM are non-reactive and do not consider motion behavior that deviates from the human operator. Therefore, the NC metric penalizes collisions classified as at-fault. Such at-fault cases are (1) collisions with a stationary bounding box, (2) ego-front collisions with any detected entity, and (3) ego-side collisions when the ego-vehicle is in an intersection or multiple lanes. The collision metric additionally distinguishes between vulnerable agents (vehicles, pedestrians, bicycles) and static classes (e.g., traffic cones, generic objects), with the score given by:

$$\texttt{score}_{\texttt{NC}} = \begin{cases} 1 & \text{no at-fault collision,} \\ 0.5 & \text{one at-fault collision with static class,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Drivable Area Compliance (DAC). Vehicles must remain in drivable areas to ensure traffic law compliance and pedestrian safety. In NAVSIM, drivable areas include lanes, intersections, or parking areas. If any corner of the ego-vehicle bounding box leaves the drivable area, the metric assigns a multiplied penalty of $\texttt{score}_{\texttt{DAC}} = 0$, or $\texttt{score}_{\texttt{DAC}} = 1$ upon compliance.

Ego Progress (EP). The ego progress metric ensures that the vehicle is advancing along the intended route as specified by the navigation goal. Specifically, the progress result $\texttt{score}_{\texttt{EP}}$ represents the agent progress as a ratio to an approximated safe upper bound. We use a search-based strategy with trajectory proposals of PDM-Closed [6], the SoTA planner on the benchmark, to identify this upper bound. The ratio is clipped to $[0, 1]$ while discarding low or negative progress scores, e.g., if the upper bound is below 5 meters.

Time to Collision (TTC). The TTC score ensures that the ego vehicle maintains a safe distance from other actors and avoids near collisions. In NAVSIM, TTC is the minimum time in any iteration until the ego vehicle collides with a bounding box. At each simulated iteration, the metric projects the ego-vehicle with a constant velocity and heading with a step size of 0.3s and checks for collisions. Certain collisions are discarded, e.g., if the bounding box is behind the ego vehicle. The result is either $\texttt{score}_{\texttt{TTC}} = 1$, if $\texttt{TTC} > 0.9$s, or $\texttt{score}_{\texttt{TTC}} = 0$ otherwise.

Comfort (C). This comfort metric verifies that several kinematic statistics, such as acceleration and jerk, are within predefined thresholds. We use the same statistics and thresholds as the nuPlan framework [13], which were determined based on the human driver. If the planner complies with all thresholds, the result is $\texttt{score}_{\texttt{C}} = 1$, otherwise $\texttt{score}_{\texttt{C}} = 0$.

Additional Subscores. We implemented additional subscores for PDMS, such as driving direction compliance and speed limit compliance. However, we observed technical issues or little impact of these subscores with their initial implementations and have excluded the metrics from the main study and inaugural challenge. We also experimented with traffic light infractions but faced difficulties with the annotations. The nuPlan dataset applies several post-processing steps on the traffic light states [13], e.g., to fill labels or to ensure consistency on intersections. While this post-processing provides complete traffic lights for all actors in the scene, we frequently encountered false traffic light infractions by the human operator and, therefore, did not include the subscore in PDMS. We aim to extend the PDMS with improved subscores in future NAVSIM competitions.

Metric Update v1.1. Following the initial NAVSIM challenge, we updated the PDMS implementation for the public `navtest` leaderboard. The update includes (1) a minor fix in the trajectory selection of

	Name	Logs	Sensors	Config parameters
Standard	`trainval`	14 GB	>2000 GB	`train_test_split=trainval`
	`test`	1 GB	217 GB	`train_test_split=test`
	`mini`	1 GB	151 GB	`train_test_split=mini`
NAVSIM	`navtrain`	-	445 GB*	`train_test_split=navtrain`
	`navtest`	-	-	`train_test_split=navtest`
	`navmini`	-	-	`train_test_split=navmini`
Leaderboard	`private_test_e2e`	<1 GB	25 GB	`train_test_split=private_test_e2e`

Table 2: **Dataset Splits.** Available data splits and storage requirements of OpenScene (i.e., standard) and the filtering options for NAVSIM. We provide standalone downloads for the leaderboard split and the `navtrain` sensors (*requiring 300GB when excluding past frames).

the PDM-Closed planner [6], leading to a different normalization constant in the ego-progress and (2) a re-implementation of the driving direction compliance (DDC) metric in nuPlan [13]. The DDC metric verifies that the distance traveled in oncoming traffic does not exceed predefined thresholds within a 1 second window. During evaluation, the DDC metric is merely collected in the background, but does not contribute to the PDMS.

3 Dataset & Data Splits

This section discusses the supported data for NAVSIM and the associated licenses. Additionally, we offer a comprehensive overview of the data splits and curation.

3.1 Dataset Support of OpenScene

NAVSIM provides initial support for the OpenScene v1.1 dataset [4], a collection of 120h driving data derived from the nuPlan dataset [13]. OpenScene reduces the temporal frequency of sensor and annotation data to 2Hz, thereby significantly reducing storage demands. As such, OpenScene is distributed under *Creative Commons Attribution Non-Commercial Share Alike 4.0* (CC BY-NC-SA 4.0) [5] and nuPlan's *Dataset License Agreement for Non-Commercial Use* [19]. The licensing covers all available splits in OpenScene, including the private data for the NAVSIM challenge. We refer to nuPlan's terms of use [19] for information on the privacy policy or takedown request regarding privacy concerns. We thank the nuPlan team from Motional for their consent to the re-distribution of OpenScene and for providing a private data split for the competition.

In the future, we aim to extend the support for more datasets in NAVSIM. Such datasets ideally provide sensor data as driving policy input, with tracked traffic entities in BEV (e.g., bounding boxes) and a semantic HD map for evaluation. We hope more datasets are openly released to the community, allowing for diverse and challenging benchmarks.

3.2 Dataset Splits & Curation in NAVSIM

Dataset Splits. We summarize all splits used in NAVSIM in Table 2. The OpenScene dataset provides several data splits, such as `trainval` for training and validating driving policies, `test` as split for regular testing, and `mini` for lightweight demonstrations (see "Standard" in Table 2). Each split consists of (1) log files for annotations, the ego status, or metadata and (2) sensor data providing surround view camera images and LiDAR point clouds. For NAVSIM, we provide fixed filtering options for challenging scenarios on the respective standard splits, called `navtrain`, `navtest`, and `navmini`. These filtering options do not require additional downloads and can be applied to the OpenScene splits. We recommend the usage of `navtrain` and `navtest` as standardized training and evaluation protocols, e.g., when benchmarking in research papers. Given the storage demands of sensors in `trainval`, we provide a separate download only for the frames needed in `navtrain`. These `navtrain` sensor frames require only 445GB of storage or 300GB when a driving agent does not require past sensor frames. Lastly, the `private_test_e2e` split serves as the private set for the HuggingFace leaderboard, with removed annotations and metadata in the log files to ensure the integrity of the challenge.

Split	Agent	NC	DAC	TTC	Comf.	EP	PDMS
`trainval`	Const. Velocity	93.1	89.7	88.8	100	73.3	78.7
	Human	99.4	97.0	97.9	99.9	84.7	90.9
`navtrain`	Const. Velocity	68.6	59.3	47.8	100	21.1	22.4
	Human	100	100	100	99.9	88.0	94.9

Table 3: **Challenging Scenes.** We show the PDMS and subscores for the constant velocity baseline and human operator on the `trainval` and `navtrain` splits.

Curation & Filtering. As mentioned in the main paper, we filter the dataset to remove frames with (1) near-trivial solutions or (2) annotation errors that provide problems during evaluation. For (1), we define trivial frames as situations where the constant velocity baseline achieves a PDMS of more than 80. We find this strategy effective in filtering frames that do not require active decision-making or intervention, as the constant velocity model (with straight driving) can be interpreted as an action-repeat policy. In (2), we remove frames where the human operator achieves a PDMS of less than 80. These failures are often due to false collisions with erroneous bounding boxes or off-road infractions, given the over-approximated ego extent. We present the PDMS results of the constant velocity agent and human operator in Table 3. The PDMS of 78.7 of the constant velocity baseline on `trainval` indicates that most frames have trivial solutions. After filtering in `navtrain`, the constant velocity PDMS drops to 22.4, whereas the human operator improves (i.e., 94.9 vs. 90.9). Thus, we ensure that the human operators provide a valid expert for imitation in terms of PDMS.

4 Baselines

In this section, we provide further details on the implementation of the baselines and experiments.

4.1 Baselines for Alignment Between Open-Loop and Closed-Loop Evaluation

Experimental Setting. To benchmark open-loop and closed-loop metrics, we use the nuPlan simulator as a reactive environment. We simulate short scenarios with a duration of $d = 15$s and frequency of $f = 10$Hz, where a planner outputs a new trajectory. As is the default in nuPlan, all planners must output a trajectory over the horizon of 8s, with positions and orientation values. The closed-loop score (CLS) is calculated over the complete simulation (i.e., $d = 15$s), whereas the open-loop score (OLS) and PDMS are computed on the first frame of each scenario. For implementation details on CLS and OLS, we refer to [13]. In the following, we outline details of the motion planners used in the main paper.

Constant Kinematics. We consider 2 constant velocity baselines when interpolating the vehicle state on the longitudinal axis of the ego vehicle or the lane centerline [6], respectively. We extend the constant velocity planner with 4 acceleration values (i.e. $\{\pm 1, \pm 2\}$ ms^{-2}) along the longitudinal ego-axis and the centerline, totaling 8 constant acceleration baselines.

IDM Planner. For the alignment study, we include the Intelligent Driver Model (IDM) [24] planner from nuPlan [13]. By default, the planner uses a fallback target velocity of $v_0 = 10\text{ms}^{-1}$, a desired gap to the leading vehicle of $s_0 = 1\text{m}$, a headway time of $T = 1.5\text{s}$, a maximum acceleration of $a = 1\text{ms}^{-2}$, a maximum deceleration (positive) of $b = 3\text{ms}^{-2}$, and a map radius of $r = 40\text{m}$. We vary the default setting and consider 2 fallback target velocities $v_0 \in \{5, 15\}\text{ms}^{-1}$, 2 desired leading gaps $s_0 \in \{0.1, 5\}\text{m}$, 2 headway times $T \in \{0, 3\}\text{s}$, 2 acceleration parameters $a \in \{4, 6\}\text{ms}^{-2}$, 2 maximum deceleration's $b \in \{4, 6\}\text{ms}^{-2}$, and 2 map radii of $r \in \{1, 10\}\text{m}$. Lastly, we add 1 'aggressive' driver profile ($v_0 = 15\text{ms}^{-1}$, $s_0 = 0.1\text{m}$, $T = 0\text{s}$, $a = 6\text{ms}^{-2}$, $b = 3\text{ms}^{-2}$) and 1 'passive' setting ($v_0 = 8\text{ms}^{-1}$, $s_0 = 5\text{m}$, $T = 3\text{s}$, $a = 1\text{ms}^{-2}$, $b = 6\text{ms}^{-2}$) to the study. With all configurations, we include a total of 15 IDM planners.

PDM-Closed. Additionally to the default PDM-Closed planner [6], which evaluates 5 target speeds $\{10, 40, 60, 80, 100\}\%$ of the speed-limit combined with 3 lateral offsets ($\{-1, 0, 1\}$m), we include a variant with only a single offset of 0m and two with only a single target speed of 100% and 200% of the speed-limit respectively. We also test one version that uses only a single sample obtained by combining an offset of 0m from the centerline with a target speed of 100% of the speed-limit. Moreover, we evaluate three versions that simulate proposals over $\{1, 2, 8\}$s instead of the default 4s. Finally, we individually drop trajectory scoring metrics for no-collision, drivable area compliance, and driving direction compliance individually and all at once. In total, this results in 12 variants of the PDM-closed planner.

Urban Driver. We consider the Urban Driver open-loop implementation [23], that is provided in nuPlan. Specifically, we train the model with 2 embedding sizes (128 and 256) on 6 datasets ($\{25\%, 50\%, 100\%\}$ of `navtrain` and an equal-sized non-filtered set) and evaluate 2 checkpoints at different epochs (i.e., after 2 and 100 epochs), resulting in a total of 24 Urban Driver models. We train all models for 100 epochs on a single NVIDIA 3090 GPU with an AdamW optimizer [18], an L_1-loss, a batch size of 64, and a learning rate of 1e^{-4} that is divided by 10 after 50 and 75 epochs. Training a single Urban Driver model takes about 1 day.

PlanCNN. For the study, we modify a PlanCNN model [22], to receive all 15 input combinations of a BEV raster, the kinematic ego status (velocity and acceleration), the navigation goal, and the centerline according to [6]. We use a ResNet-50 [9] to encode the BEV raster and apply linear layers to project all input features to a vector of size 512. We concatenate the feature vectors and apply an MLP (with 2 hidden layers and a hidden state size of 512) to regress the output trajectory. All models are trained on $\{25\%, 50\%, 100\%\}$ of `navtrain` and an equal-sized non-filtered set, resulting in 90 PlanCNN variants. We use a single NVIDIA 2080Ti GPU and train for 100 epochs with the AdamW optimizer [18], an L_1-loss function, a batch size of 64, and a learning rate of 1e^{-4} that is divided by 10 after 50 and 75 epochs. The training process takes about 1 day per model.

4.2 Baselines for End-to-End Autonomous Driving

TransFuser. We base our TransFuser [3] implementation on `carla_garage`[2], with several modifications to the original architecture. First, we stitch the front-view camera (`cam_f0`) and cropped side cameras (`cam_l0`, `cam_r0`) to a single 1024×256 input image (approx. FOV of $140°$), to replace a single wide-angle camera available in CARLA [7]. To offer a lightweight sensor baseline, we apply ResNet-34 [9] on the LiDAR and image grid. We tokenize the resulting BEV feature grid, concatenate a linear projection of the ego status (incl. velocity, acceleration, navigation goal), and forward the features as keys and values in a Transformer decoder [25]. The decoder has 3 layers and uses a dimensionality of $d_{\text{model}} = 256$ for input-output features, $d_{\text{ffn}} = 1024$ for Feed Forward Networks (FFNs), and has 8 attention heads. We use 30 learnable queries for vehicle detection and 1 learnable ego query for trajectory regression. We apply a bounding box regression head on the vehicle queries for detection according to DETR [2]. We use a simple FFN on the resulting ego query for the ego trajectory prediction. Moreover, we ignore the depth and semantic segmentation heads of TransFuser since no labels are available in NAVSIM for these tasks. However, we maintain the BEV segmentation head on the front-facing 32m $\times$ 64m to predict the rasterized drivable area, walkways, lane centerlines, static objects, vehicles, and pedestrians. We apply an L_1-loss on the ego trajectory, a Hungarian matching loss on the vehicle detection's (i.e., consisting of a cross-entropy and L_1-loss), and a cross-entropy loss on the BEV segmentation. By default, we train the TransFuser model for 100 epochs on `navtrain` with a single NVIDIA A5000 GPU, which takes about 1 day. We use an Adam optimizer [14], a batch size of 96, and a constant learning rate of 1e^{-4}. We refer to the NAVSIM repository for additional details (see Sec. 2).

Latent TransFuser (LTF). For LTF [3], we replace the LiDAR input grid with a learned tensor of the same shape. Since only a front-facing image is provided to LTF, we adapt the object detection

[2]https://github.com/autonomousvision/carla_garage

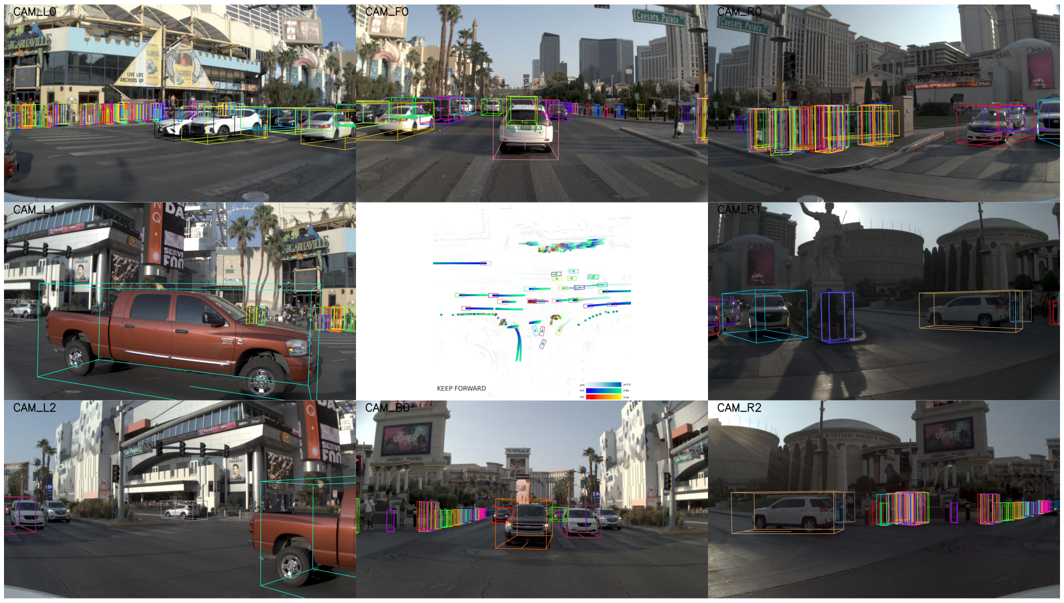

Figure 2: **Visualization of the PARA-Drive Model's output.** Eight surrounding cameras are displayed with detected object bounding boxes. In the center, the predicted BEV information is shown, including object bounding boxes, their predicted multi-modal future trajectories, as well as the planned trajectory for the ego vehicle.

auxiliary tasks only to detect bounding boxes in front of the ego vehicle (i.e., in the 32m $\times$ 64m area). The remaining training parameters and configurations are equivalent to those of TransFuser.

UniAD. We reproduce UniAD [11] using its official codebase[3]. It takes inputs from the 8 surround-view cameras, at a resolution of 1920$\times$1080 for each image. The ego status (incl. velocity and acceleration) is input to the BEV encoder. We extend the length of the temporal input frames and the trajectory planning frames to 3 and 8, respectively, to match the NAVSIM settings. The frame lengths for occupancy forecasting are also aligned. For detection, we filter the training labels based on the visibility with the LiDAR sensor. In the mapping module, we select 2 classes (pedestrian crossing and road boundary) as thing classes and 1 stuff class (i.e., drivable area). The backbone of UniAD is replaced with a ResNet-50 [9], while the other model settings remain the same. For the training strategy, we first train a 3D detection model, BEVFormer [17], on `navtrain` for better parameter initialization. This BEVFormer is initialized with a corresponding pre-trained BEVFormer on the nuScenes dataset [1]. We then train the UniAD in an end-to-end manner, combining stage 1 and stage 2 of the original UniAD training scheme. The gradient back-propagation of the image backbone is stopped to reduce memory cost. The UniAD is trained for 14 epochs with all modules, including losses for tracking, mapping, motion prediction, occupancy forecasting, and planning. The batch size is set to 1 per GPU across 96 GPUs, with a learning rate of 2e^{-4} for convergence stability. Other hyper-parameters remain unchanged. The entire training process takes 3 days.

PARA-Drive. Similar to the UniAD adaptation, we train PARA-Drive [26] following the NAVSIM settings. Both training of the UniAD and PARA-Drive share the use of the same BEVFormer pre-trained weights on the NAVSIM dataset. The primary differences between UniAD and PARA-Drive lie in the architecture design as in [26]. The batch size is set to 1 per GPU, and we use 10 nodes for training, each with 8 NVIDIA A100, with a total of 80 GPUs for about 3.5 days of training.

5 Additional Results

In this section, we provide supplementary results on the experiments of the main paper.

[3] https://github.com/OpenDriveLab/UniAD

Optimization?	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
✓	97.8	91.9	92.9	100	78.8	83.4
✗	95.4	86.0	87.3	99.3	71.9	75.2

Table 4: **UniAD runtime trajectory optimization.** We observe that trajectory optimization to minimize overlap with predicted occupancy significantly lowers PDMS on the `navtest` split.

5.1 Alignment Between Open-Loop and Closed-Loop Evaluation

For this subsection, we present additional results for the alignment study between the open- and closed-loop metrics. We consider the PDMS with an evaluation horizon of $h = 4$ and the OLS of nuPlan (as open-loop metrics) in comparison to the CLS of nuPlan [13].

Subscores. In Fig. 3, we show the correlations of subscores given the PDMS and CLS implementations. Despite the longer duration for closed-loop simulation ($d = 15$s), we observe that PMDS is able to approximate statistics across all subscores. Drivable-area compliance (DAC), ego progress (EP), or comfort (C) appear easier to approximate over the PDMS horizon of $h = 4$s, compared to the collision-based metrics (i.e. NC, TTC). However, we observe higher correlations for collision metrics with a reduced closed-loop duration ($d = 4$s). Generally, the PDMS submetrics provide valuable scores for benchmarking despite the strong assumption of a non-reactive trajectory planner.

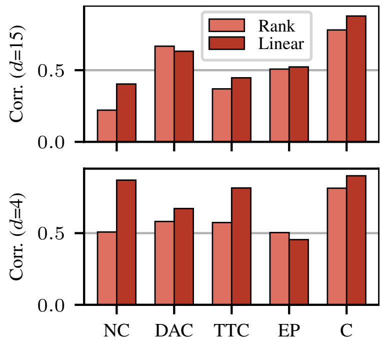

Figure 3: **Correlation of Subscores.**

Filtered Training. In Fig. 4, we show the CLS results for the learned planners from Section 4.1. We compare the box plots when training on $\{50\%, 100\%\}$ of the filtered `navtrain` split and an equal sized non-filtered split, given a planning frequency $f = 2$Hz, that yielded the highest scores for learned agents. For 100% training data, we observe minor differences in performance, where `navtrain` indicates higher variance, whereas the unfiltered split results in a wider range (indicated by the whiskers). The PlanCNN variant, which omits the ego status, achieves the highest CLS results on both training splits (i.e., 0.65 vs. 0.67). Moreover, when using 50% of training data, the planners perform worse when trained on the `navtrain` split. We conclude that training on filtered data does not necessarily benefit a learned planner but does not harm performance when given a sufficient training size.

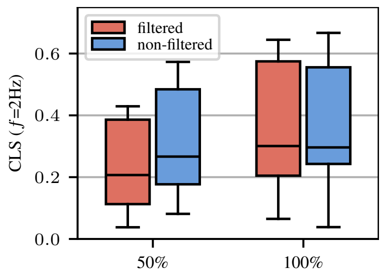

Figure 4: **Training Split CLS.**

Additionally, we present our benchmark results for privileged planning in Fig. 5, where we show the scatter plots for different CLS durations d (4s to 15s), planning frequencies f (10Hz vs. 2Hz), and evaluation horizons h of PDMS (2s to 8s).

5.2 Revisiting the State of the Art in End-to-End Autonomous Driving

Occupancy-Based Trajectory Optimization. The default implementation of UniAD uses the network's predicted occupancy to optimize the planned trajectory using Newton's method, minimizing its overlap with occupied cells in a binary occupancy grid. As shown in Table 4, this leads to a drop in performance, in particular due to more off-road driving and reduced progress. This optimization is also run independently per waypoint, leading to a slightly reduced comfort score. We disable this optimization for our results reported in the main paper.

ID	Track	Motion	Occ.	Plan	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
A	✓	✓	✓	✓	97.8	92.0	**93.7**	99.7	78.3	83.6
B	✓	✓		✓	**97.9**	**92.4**	93.0	99.8	**79.3**	**84.0**
C	✓			✓	97.1	91.5	91.5	99.7	78.2	82.3
D				✓	97.1	90.9	92.0	99.8	77.0	81.9

Table 5: **PARA-Drive ablations on the effectiveness of each task.** We assess the impact of the auxiliary tasks in PARA-Drive on the `navtest` split. The default configuration, which excludes occupancy prediction, has the highest PDMS.

Team	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
1. Hydra-MDP [16]	97.9	**100**	**95.9**	100	**87.5**	**92.7**
2. ZERON	97.2	97.9	94.5	98.6	80.0	87.5
3. xiaokt001	**98.6**	95.2	93.1	100	79.4	85.5
4. Neuros	97.9	95.2	93.1	100	79.5	85.4
5. ppp	96.6	95.2	93.8	100	78.4	85.0
Baseline: TransFuser	97.2	94.5	92.4	100	80.0	84.8

Table 6: **CVPR 2024 NAVSIM Challenge.**

Analysis of Auxiliary Tasks. We study the performance influence of auxiliary tasks for PARA-Drive in Table 5. All auxiliary tasks (A), including tracking, motion forecasting, and occupancy prediction, lead to a PDMS of 83.6. Interestingly, we observe the highest PDMS (i.e., 84.0) when excluding the occupancy prediction (B) and thus use this setting by default in the main paper. However, our results indicate the value of the motion prediction heads, with a PDMS reduction of 1.7 when left out (C). Only training for the downstream planning task, i.e., further removing tracking in (D), we observe the lowest PDMS of 81.9 on `navtest`. We conclude that auxiliary tasks are beneficial for planning in NAVSIM but emphasize that further studies are needed to improve performance across the subscores. We further visualize the PARA-Drive predictions in Fig. 2.

6 2024 NAVSIM Challenge

In this section, we present supplementary details on the 2024 NAVSIM challenge. The nuPlan team from Motional provided additional data for the competition, which we carefully processed for the leaderboard evaluation. We only release non-overlapping sequences with sensor data and ego status information to the competitors in the local coordinate frame. This prevents ground-truth pose leakage or utilization of the HD map that operates in global coordinates. Moreover, we remove future poses, annotations, and metadata from the log files and filter the dataset for challenging scenarios. The processing results in a smaller split compared to `navtest` but ensures the integrity of the challenge.

In Table 6, we present the results from the 2024 NAVSIM challenge of the top 5 teams. The competition was held from March to May 2024 and received 463 submissions from 143 teams. The winning team proposed Hydra-MDP [16], which extends TransFuser with multiple trajectory candidates, which are selected based on a predicted cost term. This cost function considers several factors, such as human similarity, collision avoidance, or on-road compliance, that are provided as targets during training. Hydra-MDP achieves top scores in almost all submetrics, with a PDMS of 92.7 on the leaderboard. Team ZERON ranked second with a PDMS of 87.5 and applied a vision language model (VLM) in their driving agent. We want to actively maintain the evaluation server to provide a standardized benchmark for diverse driving approaches. Moreover, we aim to hold future competitions with additional metrics and larger evaluation sets.

7 Broader Impact

NAVSIM provides new simulation-based metrics to evaluate driving policies from sensor data. We only consider data from three US cities and Singapore, which can lead to geographical biases in the benchmark. While the metrics in NAVSIM show advances in terms of safety assessment of autonomous vehicles, several attributes, including traffic law compliance or ethical decisions, are not considered for the benchmark. We hope our open-source framework will encourage the adoption of more principled driving metrics for research on autonomous vehicles.

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[3] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. TransFuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2023.

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

[5] Creative Commons. Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License. https://creativecommons.org/licenses/by-nc-sa/4.0. Accessed: 2024-06-12.

[6] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Proc. Conf. on Robot Learning (CoRL)*, 2023.

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017.

[8] The Apache Software Foundation. Apache license, version 2.0. http://www.apache.org/licenses/LICENSE-2.0. Accessed: 2024-06-12.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] 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 *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[11] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[12] 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 *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023.

[13] Napat Karnchanachari, Dimitris Geromichalos, Kok Seang Tan, Nanxiang Li, Christopher Eriksen, Shakiba Yaghoubi, Noushin Mehdipour, Gianmarco Bernasconi, Whye Kit Fong, Yiluan Guo, and Holger Caesar. Towards learning-based planning: The nuPlan benchmark for real-world autonomous driving. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2024.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

[15] Norman Lehtomaki, Nils Sandell, and Michael Athans. Robustness results in linear-quadratic gaussian based multivariable control designs. *IEEE Trans. on Automatic Control (TAC)*, 1981.

[16] Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, Yu-Gang Jiang, and Jose M. Alvarez. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv.org*, 2024.

[17] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

[19] Motional. Dataset License Agreement for Non-Commercial Use. https://www.nuscenes.org/terms-of-use. Accessed: 2024-06-12.

[20] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2019.

[21] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[22] Katrin Renz, Kashyap Chitta, Otniel-Bogdan Mercea, Sophia Koepke, Zeynep Akata, and Andreas Geiger. Plant: Explainable planning transformers via object-level representations. In *Proc. Conf. on Robot Learning (CoRL)*, 2022.

[23] Oliver Scheel, Luca Bergamini, Maciej Wolczyk, Błażej Osiński, and Peter Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Proc. Conf. on Robot Learning (CoRL)*, 2021.

[24] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 2000.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[26] Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[27] 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.org*, 2305.10430, 2023.

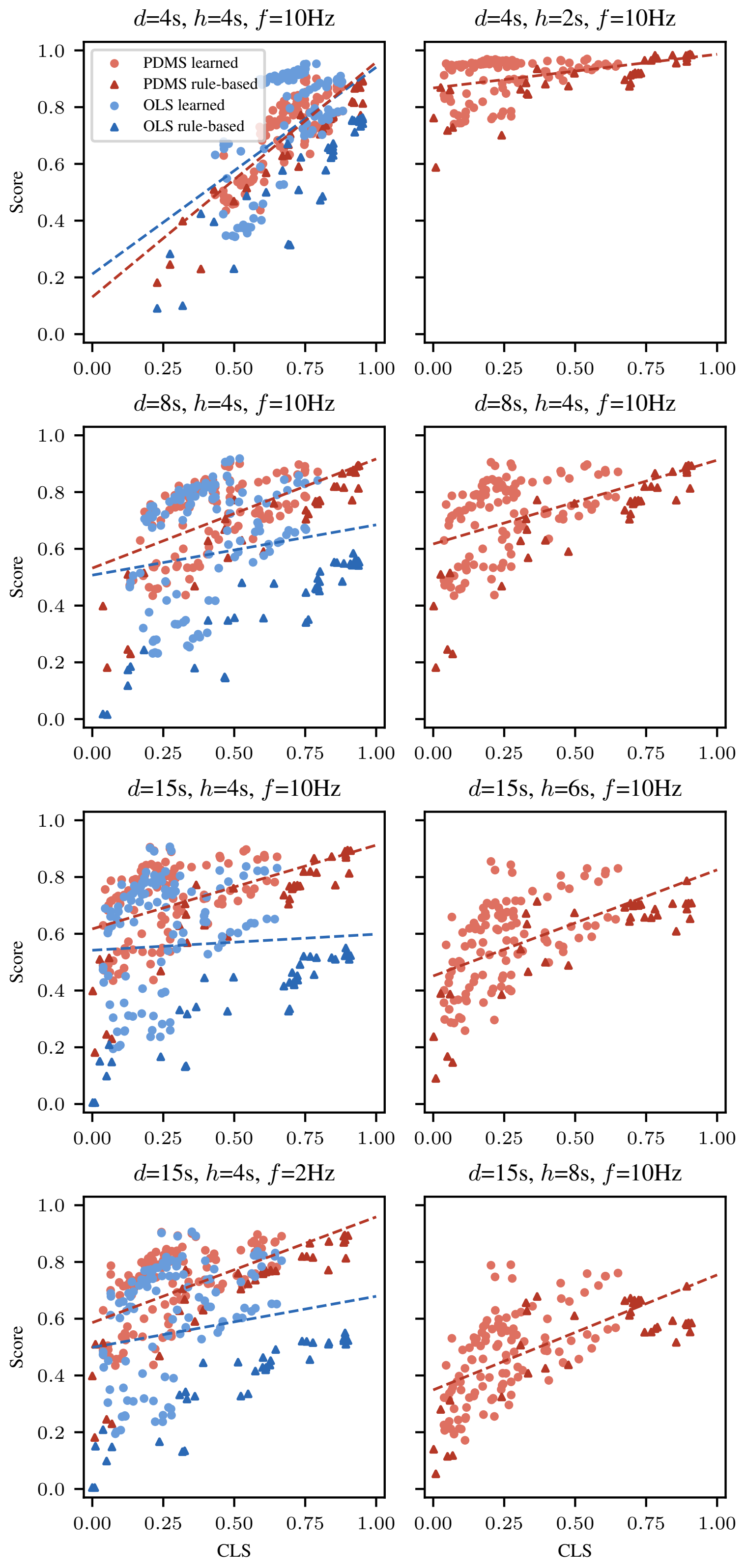

Figure 5: **Closed-Loop Alignment.** We visualize the open-loop metrics (OLS, PDMS) against the closed-loop score (CLS) of the privileged planners. On the left column, we vary the CLS duration d (4s to 15s) and planning frequency f (10Hz vs. 2Hz). The right column shows different PDMS horizons h (2s to 8s). (Default: $d = 15$s, $h = 4$s, $f = 10$Hz).