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: Benchmarking vision-based driving policies is challenging. On one hand, open-loop evaluation with real data is easy, but these results do not reflect closed-loop performance. On the other, closed-loop evaluation is possible in simulation, but is hard to scale due to its significant computational demands. Further, the simulators available today exhibit a large domain gap to real data. This has resulted in an inability to draw clear conclusions from the rapidly growing body of research on end-to-end autonomous driving. In this paper, we present NAVSIM, a middle ground between these evaluation paradigms, where we use large datasets in combination with a non-reactive simulator to enable large-scale real-world benchmarking. Specifically, we gather simulation-based metrics, such as progress and time to collision, by unrolling bird's eye view abstractions of the test scenes for a short simulation horizon. Our simulation is non-reactive, *i.e.*, the evaluated policy and environment do not influence each other. As we demonstrate empirically, this decoupling allows open-loop metric computation while being better aligned with closed-loop evaluations than traditional displacement errors. NAVSIM enables to benchmark driving policies on a large set of challenging scenarios, resulting in several new insights. We observe that simple methods with moderate compute requirements such as TransFuser can match recent large-scale end-to-end driving architectures such as UniAD. Our framework can potentially be extended with new datasets, data curation strategies, and metrics, and will be continually maintained. Our code is available at https://github.com/autonomousvision/navsim.

1 Introduction

Autonomous vehicles (AVs) have gained immense research interest due to their potential to change transportation and improve traffic safety [1, 2]. This has created a large community working on the development of AV algorithms, which map high-dimensional sensor data to desired vehicle control outputs. Therefore, measuring and comparing the performance of AV algorithms is a crucial task.

Unfortunately, it is extremely challenging to evaluate driving performance, and the most widely-used benchmarks today fall short in several respects: (1) the datasets used, such as nuScenes [3], were created for perception tasks such as object detection. As such, they focus on visual diversity and label quality instead of the relevance of the data for research on planning. Often, most frames have a trivial solution of extrapolating the historical driving behavior, leading to "blind" driving policies that observe only the vehicle's past trajectory obtaining state-of-the-art performance [4, 5, 6]. (2) Due to the fact that driving is an inherently multifaceted task where the algorithm must coordinate several desired properties such as safety, comfort, and progress, the evaluation metric must also balance potentially conflicting objectives. However, as shown in Fig. 1, existing metrics such as the average displacement error (ADE) between a predicted and recorded human trajectory often misrepresent the relative accuracy of trajectories. (3) Since driving involves interactions among multiple agents,

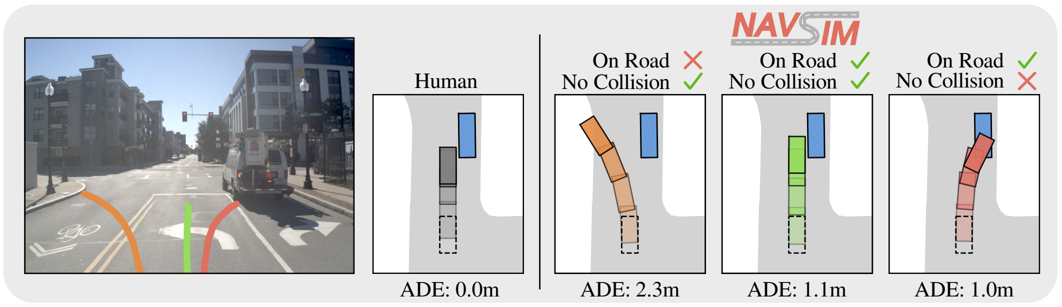

Figure 1: **NAVSIM.** Traditional metrics such as the average displacement error (ADE) overlook the multi-modality of driving. They penalize trajectories that deviate from a recorded human driving log, even if such a trajectory is safe. Our benchmark evaluates trajectory outputs of sensor-based driving policies with simulation-based metrics, considering collisions and map compliance.

evaluation must ideally be interactive, e.g., in simulation. Unfortunately, existing simulators with synthetic sensor data exhibit a significant domain gap to real-world driving. (4) Besides, the lack of a standardized evaluation setup has led to subtle inconsistencies between metrics in existing work, leading to unfair comparisons and inaccurate conclusions [5, 7]. Collectively, these problems hinder progress in the development of AVs, emphasizing the need for more principled benchmarks.

In this work, we take steps towards alleviating these issues. First, we propose a strategy for sampling interesting driving scenarios and apply it to the largest publicly-available driving dataset [8]. We obtain, for the first time, over 100k challenging real-world driving scenarios for training and evaluating sensor-based driving policies. We show that in these scenarios, "blind" driving policies fail to compete with more principled sensor-based policies. Second, we draw inspiration from the literature of rule-based planning for AVs [6, 9, 10, 11] to identify a set of diverse, efficient, and principled metrics that cover multiple facets of the autonomous driving task. Third, we circumvent the need for inaccurate sensor simulation with domain gaps by simplifying our simulation to a non-reactive one. Given an observed real-world sensor input, the agent under test commits to a set of actions for a specific time horizon. Further, these actions are assumed to not affect the future behavior of other agents in the scene. Under this setting, it is possible to simulate the expected motion of all agents over this time horizon in a simplified bird's-eye-view (BEV) abstraction of the scene, and incorporate metrics that involve interactions, as we observe in Fig. 1. Empirically, we demonstrate that our selected metrics are well-correlated to the outcomes of closed-loop simulations.

We combine these ideas to propose NAVSIM, a comprehensive tool for AV data curation, simulation, and benchmarking. We instantiate standardized training and evaluation splits for NAVSIM with the OpenScene dataset [12], though our framework can be extended to other datasets. With these splits, we present a detailed analysis of popular end-to-end driving models previously benchmarked either exclusively on CARLA [13] or nuScenes [3], providing the first direct comparison between these families of approaches in an independent evaluation setting. Interestingly, we find that the performances of the best methods developed in both settings are similar, despite a vast difference in computational requirements for their training.

Contributions. (1) We build NAVSIM, a framework for non-reactive AV simulation, with standardized protocols for training and testing, data curation tools ensuring broad accessibility. (2) We develop configurable simulation-based metrics that are well-suited for evaluating sensor-based motion planning. (3) We reimplement a collection of end-to-end approaches for NAVSIM including TransFuser, UniAD, and PARA-Drive, showcasing the surprising potential of simple models in our challenging scenarios.

2 Related Work

End-to-End Driving. End-to-end driving streamlines the entire stack from perception to planning into a single optimizable network. This eliminates the need for manually designing intermediate representations. Following pioneering work [14, 15, 16], a diverse landscape of end-to-end models has emerged. For instance, an extensive body of end-to-end approaches focuses on closed-loop simulators, utilizing single-frame cameras, LiDAR point clouds, or a combination of both for expert imitation [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]. More recently, developing end-to-end models on open-loop benchmarks has gained traction [5, 7, 28, 29, 30, 31]. Our work introduces a new evaluation scheme with which we compare end-to-end models from both communities.

Closed-Loop Benchmarking with Simulation. Driving simulators allow us to evaluate autonomous systems in a closed-loop manner and collect downstream driving statistics, including collision rates, traffic-rule compliance, or comfort. A broad body of research conducts evaluations in simulators, such as CARLA [13] or Metadrive [32] with sensor simulation, or nuPlan [8] and Waymax [33] for data-driven simulation. Unfortunately, ensuring realism when simulating traffic behavior or sensor data remains a challenging task. To simulate camera or LiDAR sensors, most established simulators rely on graphics-based rendering methods, leading to an inherent domain gap in terms of visual fidelity and sensor characteristics. Data-driven simulators for motion planning incorporate traffic recordings but do not support image or LiDAR-based methods [8, 33, 34]. Data-driven sensor simulation leverages and adapts real-world sensor data to create new simulations where the vehicle may move differently, but the rendering quality of existing tools is subpar [35, 36, 37]. Further, while promising image [38] or LiDAR [39] synthesis approaches exist, efficiently simulating sensors entirely from data remains an open problem. In this work, we provide an approach for the evaluation of real sensor data with simulation-based metrics by making a simplifying assumption that the agent and environment do not influence each other over a short simulation horizon. Despite this strong assumption, when benchmarking on real data, NAVSIM better reflects planning performance than established evaluation protocols, as demonstrated through our systematic experimental analysis.

Open-Loop Benchmarking with Displacement Errors. Open-loop evaluation protocols commonly measure displacement errors between trajectories of a recorded expert (i.e., of a human driver) and a motion planner. However, several issues concerning evaluation with displacement errors have surfaced recently, particularly on the nuScenes dataset [3]. Given that nuScenes does not provide standardized planning metrics, prior work relied on independent implementations, which led to inconsistencies when reporting or comparing results [5, 7]. Next, most planning models in nuScenes receive the human trajectory endpoint as a discrete direction command [5, 7, 28, 29, 30], thereby leaking ground-truth information into inputs. Moreover, about 75% of the scenarios in nuScenes involve straight driving [5], leading to simple solutions when extrapolating the ego-motion. For instance, AD-MLP demonstrates that an MLP on the kinematic ego status (ignoring perception completely) can achieve state-of-the-art displacement errors [4]. Such blind agents are undeniably dangerous, which highlights a broader concern: displacement metrics are not correlated to closed-loop driving [6, 13, 40, 41]. In this work, we address prevalent issues of nuScenes and propose a standardized driving benchmark with challenging scenarios and an official evaluation server. We derive a navigation goal from the lane graph instead of the human trajectory to prevent label leakage, and propose principled simulation-based metrics as an alternative to displacement errors.

3 NAVSIM: Non-Reactive Autonomous Vehicle Simulation

NAVSIM combines the ease of use of open-loop benchmarks such as nuScenes [3] with metrics based on closed-loop simulators such as nuPlan [8]. In the following, we give a detailed introduction to the task and metrics that driving agents are challenged with in NAVSIM. Subsequently, we propose a filtering method to obtain standardized train and test splits covering challenging scenes.

Task description. Driving agents in NAVSIM must plan a trajectory, defined as a sequence of future poses, over a horizon of h seconds. Their input contains streams of *past* frames from onboard sensors, such as cameras, LiDAR, as well as the vehicle's current speed, acceleration, and navigation goal, jointly termed the ego status. For compatibility with prior work [7, 28, 29, 30], we provide the navigation goal as a one-hot vector with three categories: left, straight, or right.

Non-Reactive Simulation. Traditional closed-loop benchmarks normally infer planners at high frequencies (e.g., 10Hz) [8, 13]. However, this requires efficient simulation of all input modalities of the driving agent, including high-dimensional sensor streams in the case of sensor-based approaches. To sidestep this, the core idea of NAVSIM is to evaluate driving agents using a non-reactive simulation. This means driving agents are only employed in the initial frame of each scene. Afterwards, the planned trajectory is kept fixed for the entire trajectory duration. Over this short horizon, no environmental feedback is provided to the driving agent, and the NAVSIM evaluation is purely based on the initial real-world sensor sample. This makes the agent's task more challenging, limiting simulations to short horizons. We select a horizon of $h = 4$ seconds, which has been shown in prior work to be adequate for closed-loop planning [6]. Despite this limitation, non-reactive simulation offers a key advantage: unlike traditional open-loop benchmarks, which mainly compare the planned trajectory to the human driver's trajectory in a similar setting, it enables the use of simulation outcomes to compute metrics reflecting safety, comfort, and progress. An LQR controller [42] is applied at each simulation iteration to calculate steering and acceleration values, and a kinematic bicycle model [43] propagates the ego vehicle. We execute this pipeline at 10Hz over the 4s trajectory horizon. In Sec. 4.1, we show that despite our simplifying assumption, our evaluation results in a much better alignment with closed-loop metrics than traditional open-loop metrics achieve.

PDM Score. NAVSIM scores driving agents in two steps. First, subscores in range $[0, 1]$ are computed after simulation. Second, these subscores are aggregated into the PDM Score (PDMS) $\in [0, 1]$. It is named after the Predictive Driver Model (PDM) [6], a state-of-the-art rule-based planner which uses this scoring function to evaluate trajectory proposals during closed-loop simulation in nuPlan. The metric is also an efficient reimplementation of the nuPlan closed-loop score metric [8]. In NAVSIM, the PDMS can be adapted by adding or removing subscores, changing aggregation parameters, or making subscores more challenging, e.g., by adapting their internal thresholds. It is calculated per frame and averaged across frames. In this work, we use the following aggregation of subscores:

$$\text{PDMS} = \underbrace{\left(\prod_{m \in \{\texttt{NC},\texttt{DAC}\}} \texttt{score}_m \right)}_{\text{penalties}} \times \underbrace{\left(\frac{\sum_{w \in \{\texttt{EP},\texttt{TTC},\texttt{C}\}} \texttt{weight}_w \times \texttt{score}_w}{\sum_{w \in \{\texttt{EP},\texttt{TTC},\texttt{C}\}} \texttt{weight}_w} \right)}_{\text{weighted average}}. \tag{1}$$

Subscores are categorized by their importance as penalties or terms in a weighted average. A penalty punishes inadmissible behavior such as collisions with a factor < 1. The weighted average aggregates subscores for other objectives such as progress and comfort. In the following, we briefly describe each subscore. More details can be found in Appendix B.

Penalties. Avoiding collisions and staying on the road is imperative for motion planning as it ensures traffic rule compliance and the safety of pedestrians and road users. Thus, failing to drive with no collisions (NC) with road users (vehicles, pedestrians, and bicycles) or infractions with regard to drivable area compliance (DAC) result in hard penalties of $\texttt{score}_{\texttt{NC}} = 0$ or $\texttt{score}_{\texttt{DAC}} = 0$ respectively. This results in a PDMS of 0 for the current scene. We ignore certain collisions that are not considered "at-fault" in the non-reactive environment, e.g. when the ego vehicle is static. For collisions with static objects, we apply a softer penalty of $\texttt{score}_{\texttt{NC}} = 0.5$.

Weighted Average. The weighted average accounts for ego progress (EP), time-to-collision (TTC), and comfort (C). The ego progress subscore $\texttt{score}_{\texttt{EP}}$ represents the agent progress along the route center as a ratio to an approximated safe upper bound from the PDM-Closed planner [6]. PDM-Closed obtains a possible progress value without collisions or off-road driving with a search-based strategy based on trajectory proposals. The final ratio is clipped to $[0, 1]$ while discarding low or negative progress scores if the upper bound is below 5 meters. Next, the TTC subscore ensures that driving agents respect the safety margins to other vehicles. Defaulting to a value of 1, this subscore is set to 0 if for any simulation step within the 4s horizon, the ego-vehicle's time-to-collison, when projected forward with a constant velocity and heading, is less than a certain threshold. Finally, the comfort

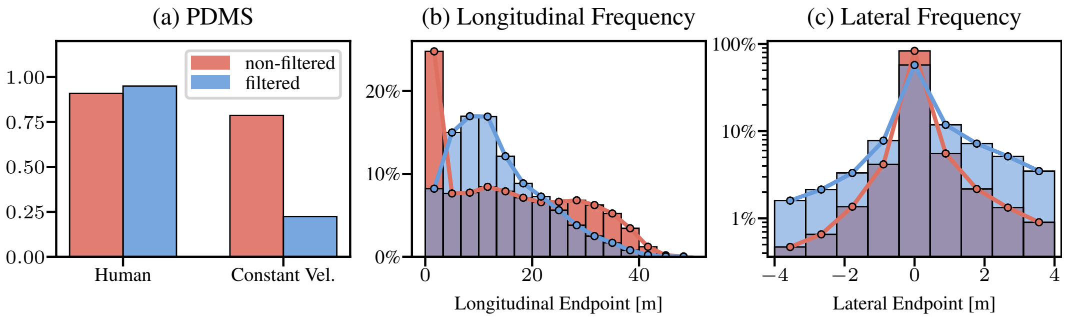

Figure 2: **Filtering.** (a) We consider challenging scenes where maintaining a constant velocity and heading fails compared to the human driver. (b) Our filtering primarily removes scenes with static or fast longitudinal movement and (c) leads to more diversity in lateral movement (log-scale).

subscore is obtained by comparing the acceleration and jerk of the trajectory to predetermined thresholds. Following the cost weights used by the PDM-Closed planner and the 2023 nuPlan challenge, we set the coefficients of the weighted average as $\texttt{weight}_{\texttt{EP}} = 5$, $\texttt{weight}_{\texttt{TTC}} = 5$, and $\texttt{weight}_{\texttt{C}} = 2$. We find this selection reasonable and robust to changes.

3.1 Generating Standardized and Challenging Train and Test Splits

Dataset. The NAVSIM framework is agnostic to the choice of driving dataset. We choose OpenScene [12], a redistribution of nuPlan [8], the largest annotated public driving dataset. OpenScene includes 120 hours of driving at a reduced frequency of 2Hz typically considered by end-to-end planning algorithms, resulting in a $10\times$ reduction of data storage requirements compared to nuPlan from over 20 TB to 2 TB. Our agent input, based on OpenScene, comprises eight cameras, each with a resolution of 1920×1080 pixels, and a merged LiDAR point cloud from five sensors. The input includes the current time-step and optionally 3 past frames, totaling 1.5s at 2Hz. In principle, any driving dataset that provides annotated HD maps, object bounding boxes, and sensor data can be converted into this format and thus be used with NAVSIM.

Filtering for challenging scenes. A majority of human driving data involves trivial situations such as being stationary or straight driving at a near constant speed. These can be solved efficiently by simple heuristics, e.g., as depicted in Fig. 2 (a), the baseline of maintaining a constant velocity and heading achieves a PDMS of 79% on the OpenScene dataset, where human-level performance corresponds to 91%. In NAVSIM, we propose the use of a filtered dataset to remove frames with (1) near-trivial solutions and (2) significant annotation errors. We remove highly simplistic scenes by detecting if the previously mentioned constant velocity agent exceeds a PDMS of 0.8. Similarly, we remove scenes in which the human trajectory results in a PDMS of less than 0.8. This ensures that an acceptable solution exists to these difficult scenarios and filters out noisy annotations such as inaccurate bounding boxes. These thresholds can be adjusted based on the desired filtered dataset size. The resulting scenarios are challenging, which is underlined by the score of the constant velocity agent dropping to 22%, whereas the human expert achieves a score of 95%. The higher ratio of non-trivial scenarios, such as turning, also results in endpoints being less distant longitudinally when nonzero, and more evenly distributed laterally, as seen in Fig. 2 (b-c). We employ this filtering strategy to provide standardized splits for training and testing, called `navtrain` and `navtest`, with 103k and 12k samples respectively. This curated data serves as a benchmark accessible as a standalone download option with a moderate storage demand given its large scale and diversity (450 GB).

4 Experiments

In this section, we present the results of our experiments aimed at answering the following questions: (1) Can non-reactive open-loop simulation provide sufficient correlation to closed-loop metrics? (2) What new conclusions do experiments on NAVSIM provide compared to prior benchmarks?

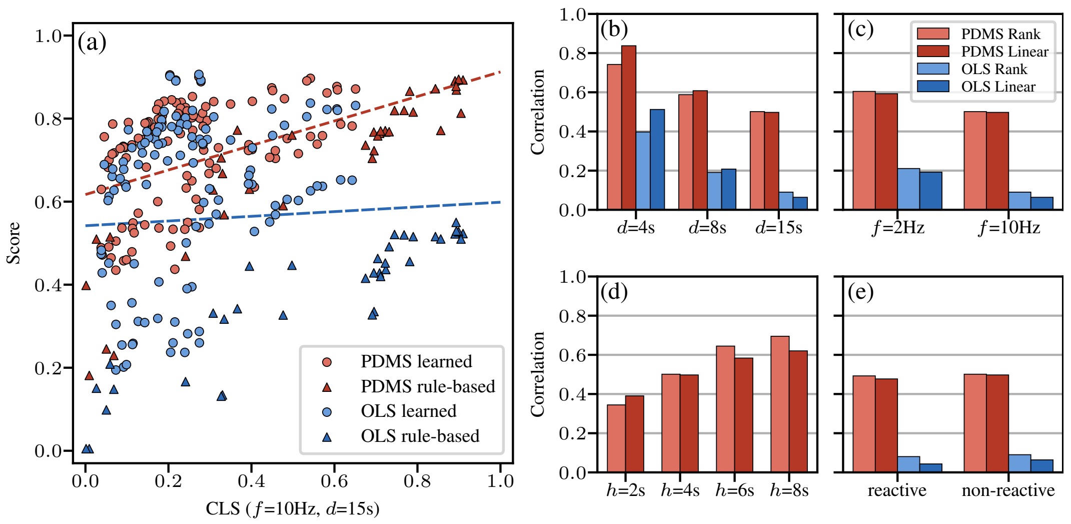

Figure 3: **Closed-Loop Alignment.** (a) For each planner, we show open-loop metrics (OLS, PDMS) together with the corresponding closed-loop score (CLS). The trendlines depicting correlations are fit linearly to all (learned and rule-based) planners. Moreover, we analyze different (b) CLS durations d, (c) planning frequencies f, (d) PDMS horizons h, and (e) closed-loop background agent behaviors.

4.1 Alignment Between Open-Loop and Closed-Loop Evaluation

Open-loop metrics should ideally be aligned with closed-loop metrics in their evaluation of different driving algorithms. In this section, we benchmark a large set of planners to analyze the alignment of closed-loop metrics with traditional distance-based open-loop metrics and the proposed PDMS.

Benchmark. Studying the relation of closed-loop and open-loop metrics necessitates access to a fully reactive simulator. To stay compatible with the dataset, we use the nuPlan simulator [8], which enables simulation for privileged planners with access to ground-truth perception and HD map inputs. Similar to PDMS, nuPlan combines weighted averages and multiplied penalties in two official scores: the **open-loop score (OLS)** aggregates displacement and heading errors with a multiplied miss-rate, and the **closed-loop score (CLS)** implements similar metrics from Section 3. Including PDMS, all metrics are in $[0, 1]$ with higher scores indicating better performance.

Due to the heavy computational requirements of closed-loop simulation, we evaluate on the `navmini` split. This is a new split we create for rapid testing, with 396 scenarios in total that are independent of both `navtrain` and `navtest` but filtered using the same strategy (Section 3.1) and hence similarly distributed. We note that nuPlan offers two kinds of background agents: reactive agents along lane centers based on the Intelligent Driver Model (IDM) [44], and non-reactive agents replayed from the dataset, which we employ unless otherwise stated. While reactive simulations of longer or dynamic lengths are generally desirable, e.g. to evaluate long-term decisions, enabling this requires dedicated solutions to long-horizon simulation that are not currently available in nuPlan [34]. Therefore, we default to a fixed closed-loop simulation duration of $d = 15$s, and a planning frequency of $f = 10$Hz, which are the standard closed-loop simulation settings in nuPlan [8].

Motion Planners. Open-loop metrics favor learned planners while rule-based approaches perform well in closed-loop evaluation in nuPlan [6]. We use a combination of both planner types in this experiment to cover different performance levels. In total, we include 37 rule-based planners with 2 constant velocity and 8 constant acceleration models, 15 IDM planners [44], and 12 PDM-Closed variants [6] which differ in hyperparameters for trajectory generation. For learned planning, we evaluate Urban Driver models [45] of 2 model sizes and 2 training lengths, and PlanCNN [46] models with 15 input combinations of the BEV raster, ego status, centerline, and navigation goal. We train all models on $\{25\%, 50\%, 100\%\}$ of `navtrain` and an equally sized uniformly sampled subset of OpenScene, giving 114 learned planners. See Appendix D.1 material for additional details.

Method	Ego Stat.	Image	LiDAR	Video	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
Constant Velocity	✓				68.0	57.8	50.0	100	19.4	20.6
Ego Status MLP	✓				93.0	77.3	83.6	100	62.8	65.6
LTF [24]	✓	✓			97.4	**92.8**	92.4	100	79.0	83.8
TransFuser [24]	✓	✓	✓		97.7	**92.8**	92.8	100	79.2	**84.0**
UniAD [29]	✓	✓		✓	97.8	91.9	92.9	100	78.8	83.4
PARA-Drive [7]	✓	✓		✓	**97.9**	92.4	**93.0**	99.8	**79.3**	**84.0**
Human					*100*	*100*	*100*	*99.9*	*87.5*	*94.8*

Table 1: **Navtest Benchmark.** We show the no at-fault collision (NC), drivable area compliance (DAC), time-to-collision (TTC), comfort (Comf.), and ego progress (EP) subscores, and the PDM Score (PDMS), as percentages. Relying on the ego status is insufficient for competitive results. While sensor agents improve, the gap to human performance highlights our benchmark's challenges.

Results. The alignment between metrics is presented in Fig. 3 (a-e). Compared to OLS, we consistently observe better closed-loop correlation for PDMS, in terms of Spearman's (rank) and Pearson's (linear) correlation coefficients. As shown in (a), PDMS can capture the closed-loop properties of both learned and rule-based planners, whereas distance-based open-loop metrics show a clear misalignment. Decreasing the CLS duration in (b) from $d = 15$s to $d = 4$s further raises the correlation of PDMS and OLS, as the simulation horizon more closely matches the open-loop counterparts. Interestingly, we observe a higher correlation of open-loop metrics in (c) when reducing the planning frequency to 2Hz. We expect a lower planning frequency to mitigate cumulative errors and enhance the controller's stability in simulation, leading to more precise trajectory execution. Moreover, we observe an increase in correlation for longer PDMS horizons in (d), ranging from $h = 2$s to $h = 8$s. While predicting the future motion over 8s is challenging in uncertain scenarios, our results indicate the value of long horizons when evaluating motion planners. Lastly, replacing the non-reactive background agents with reactive IDM vehicles during closed-loop simulation in (e) has little effect on the correlation, possibly due to the similar difficulty of both tasks [6]. We present additional results on the metric alignment in Appendix E.1.

4.2 Analysis of the State of the Art in End-to-End Autonomous Driving

In this section, we benchmark a collection of end-to-end architectures, which previously achieved state-of-the-art performance on existing open- or closed-loop benchmarks.

Methods. As a lower bound, we consider the **(1) Constant Velocity** baseline detailed in Section 3.1. We include an **(2) Ego Status MLP** as a second "blind" agent, which leverages an MLP for trajectory prediction given only the ego velocity, acceleration and navigation goal. As an established architecture on CARLA, we evaluate our reimplementation of **(3) TransFuser** [24], which uses three cropped and downscaled forward-facing cameras, concatenated into a 1024×256 image, and a rasterized BEV LiDAR input for predicting waypoints. It performs 3D object detection and BEV semantic segmentation as auxiliary tasks. We then consider **(4) Latent TransFuser (LTF)** [24], which shares the same architecture as TransFuser but replaces the LiDAR input with a learned embedding, hence requiring only camera inputs. Moreover, we provide two state-of-the-art end-to-end architectures for open-loop trajectory prediction on nuScenes. **(5) UniAD** [29] incorporates a wide range of tasks, such as mapping, tracking, motion, and occupancy prediction in a semi-sequential architecture, which processes feature representations through several transformer decoders culminating in a trajectory planning module. **(6) PARA-Drive** [7] uses the same auxiliary tasks, but parallelizes the network architecture, and the auxiliary task heads are trained in parallel with a shared encoder. Both UniAD and PARA-Drive use a BEVFormer backbone [47], which encodes the eight surround-view 1920×1080 camera images over four temporal frames into a BEV feature representation. Implementation details for all methods are provided in Appendix D.2.

Results. We show our results on `navtest` in Table 1. The Constant Velocity model is a lower bound, as the agent is used to identify trivial driving scenes excluded from the benchmark. The Ego Status MLP achieves a PDMS of 65.6, showing the value of the acceleration and navigation goal for avoiding collisions and driving off-road. However, we observe a clear gap between agents relying solely on the ego status and those considering sensor data, in contrast to results on nuScenes [5]. All sensor agents achieve a PDMS of over 83, where TransFuser and PARA-Drive marginally perform best, with a PDMS of 84.0. Surprisingly, the camera-only LTF achieves similar results (83.8). UniAD reaches a PDMS of 83.4, which, together with PARA-Drive, do not surpass the performance of TransFuser and LTF, despite the need for more demanding training, e.g., 80 GPUs for 3 days to train PARA-Drive versus 1 GPU for 1 day for TransFuser on the `navtrain` split. Due to the definition of at-fault collisions, which discard certain rear-collisions into the ego vehicle, we suspect that surround-view cameras used by UniAD and PARA-Drive, and LiDAR input of TransFuser, are less important than the wide-angle front camera which is the only input of LTF. The 10 PDMS discrepancy to the human operator demonstrates that `navtest` poses challenges even to well-studied end-to-end architectures. Specifically, the drivable area compliance (DAC) and ego progress (EP) subscores remain the most challenging. Notably, EP cannot be solved purely by human imitation, given that the maximum progress estimate used for normalization is based on a privileged rule-based motion planner. Interestingly, all agents achieve near-perfect comfort scores, indicating that smooth acceleration and jerk profiles are learned naturally from human imitation. We refer to Appendix E.2 for additional experiments and ablations of the end-to-end driving methods.

5 Discussion

We present NAVSIM, a framework for non-reactive AV simulation. We address several shortcomings of existing driving benchmarks and propose standardized but configurable simulation-based metrics for benchmarking driving policies. For accessibility, we provide downloadable challenging scenario splits and simple data curation methods. We show that our evaluation protocol is better aligned to closed-loop driving and benchmark an established set of end-to-end planning baselines. We hope that NAVSIM can serve as an accessible toolkit for AV researchers that bridges the gap between simulated and real-world driving.

Need for Reactive Simulation. While we show improvements over displacement errors, several aspects of driving remain unaddressed by evaluation in NAVSIM. A high PDMS does not always imply a high CLS, since our framework does not consider reactiveness or the compounding accumulation of errors in closed-loop simulation. Moreover, as in CLS, rear-end collisions into the ego vehicle are currently not classified as "at-fault", resulting in little importance given to the scene behind the vehicle in NAVSIM. In the future, data-driven sensor or traffic simulation could alleviate these issues, once such methods mature and become computationally tractable. Given these limitations of the current framework, we strongly encourage the use of graphics-based closed-loop simulators, such as CARLA [13], as complementary benchmarks to NAVSIM when developing planning algorithms.

Simplicity of Metrics. As a starting point, NAVSIM offers both interpretable open-loop subscores and a scalarizing function, which lets us provide a final score and ranking of competing approaches. In the future, multi-objective evaluation and other aggregation functions might be required. Moreover, closed-loop metrics also face problems, i.e., PDMS inherits several weaknesses of nuPlan's CLS. Both scores do not regard certain traffic rules (e.g., stop-sign or traffic light compliance) or concepts such as transit and fuel efficiency. We aim to improve the subscore definitions (e.g. the at-fault collision logic) and add more subscores during aggregation.

Call for Datasets. Certain limitations of the nuPlan dataset persist in NAVSIM, such as missing classes in the label space, minor errors in camera parameters, or noise in poses and 3D annotations. Our analysis might favor methods that are robust to such inconsistencies. In addition, the lack of road elevation data in our representation presents a challenge for integrating scenarios based on 3D annotations. We aim to support more datasets in the future, and advocate for more open dataset releases by the community for accelerating progress in autonomous driving.

Acknowledgments

This work was supported by the ERC Starting Grant LEGO-3D (850533), the DFG EXC number 2064/1 - project number 390727645, the German Federal Ministry of Education and Research: Tübingen AI Center, FKZ: 01IS18039A and the German Federal Ministry for Economic Affairs and Climate Action within the project NXT GEN AI METHODS. We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Daniel Dauner and Kashyap Chitta. We also thank HuggingFace for hosting our evaluation servers, the team members of OpenDriveLab for their organizational support, as well as Napat Karnchanachari and his team from Motional for open-sourcing their dataset and providing us the private test split used in the 2024 NAVSIM Challenge.

References

[1] J. Janai, F. Güney, A. Behl, and A. Geiger. *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*, volume 12. Foundations and Trends in Computer Graphics and Vision, 2020.

[2] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li. End-to-end autonomous driving: Challenges and frontiers. *arXiv.org*, 2306.16927, 2023.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[4] J.-T. Zhai, Z. Feng, J. Du, Y. Mao, J.-J. Liu, Z. Tan, Y. Zhang, X. Ye, and J. Wang. Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. *arXiv.org*, 2305.10430, 2023.

[5] Z. Li, Z. Yu, S. Lan, J. Li, J. Kautz, T. Lu, and J. M. Alvarez. Is ego status all you need for open-loop end-to-end autonomous driving? In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[6] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Proc. Conf. on Robot Learning (CoRL)*, 2023.

[7] X. Weng, B. Ivanovic, Y. Wang, Y. Wang, and M. Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[8] N. Karnchanachari, D. Geromichalos, K. Seang Tan, N. Li, C. Eriksen, S. Yaghoubi, N. Mehdipour, G. Bernasconi, W. Kit Fong, Y. Guo, and H. Caesar. Towards learning-based planning: The nuPlan benchmark for real-world autonomous driving. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2024.

[9] A. Sauer, N. Savinov, and A. Geiger. Conditional affordance learning for driving in urban environments. In *Proc. Conf. on Robot Learning (CoRL)*, 2018.

[10] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong. Baidu apollo EM motion planner. *arXiv.org*, 1807.08048, 2018.

[11] A. Sadat, M. Ren, A. Pokrovsky, Y. Lin, E. Yumer, and R. Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2019.

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

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017.

[14] D. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1988.

[15] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *arXiv.org*, 1604.07316, 2016.

[16] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. M. Allen, V. D. Lam, A. Bewley, and A. Shah. Learning to drive in a day. *arXiv.org*, abs/1807.00412, 2018.

[17] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Proc. Conf. on Robot Learning (CoRL)*, 2019.

[18] K. Chitta, A. Prakash, and A. Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021.

[19] A. Prakash, K. Chitta, and A. Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[20] D. Chen, V. Koltun, and P. Krähenbühl. Learning to drive from a world on rails. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021.

[21] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[22] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Proc. Conf. on Robot Learning (CoRL)*, 2022.

[23] H. Shao, L. Wang, R. Chen, S. L. Waslander, H. Li, and Y. Liu. Reasonnet: End-to-end driving with temporal and global reasoning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[24] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger. TransFuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2023.

[25] X. Jia, P. Wu, L. Chen, J. Xie, C. He, J. Yan, and H. Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[26] J. Zhang, Z. Huang, and E. Ohn-Bar. Coaching a teachable student. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[27] B. Jaeger, K. Chitta, and A. Geiger. Hidden biases of end-to-end driving models. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023.

[28] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. 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.

[29] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li. Planning-oriented autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[30] B. Jiang, S. Chen, Q. Xu, B. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang. VAD: Vectorized scene representation for efficient autonomous driving. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023.

[31] T. Ye, W. Jing, C. Hu, S. Huang, L. Gao, F. Li, J. Wang, K. Guo, W. Xiao, W. Mao, et al. Fusionad: Multi-modality fusion for prediction and planning tasks of autonomous driving. *arXiv.org*, 2023.

[32] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 45(3):3461–3475, 2022.

[33] C. Gulino, J. Fu, W. Luo, G. Tucker, E. Bronstein, Y. Lu, J. Harb, X. Pan, Y. Wang, X. Chen, J. D. Co-Reyes, R. Agarwal, R. Roelofs, Y. Lu, N. Montali, P. Mougin, Z. Yang, B. White, A. Faust, R. McAllister, D. Anguelov, and B. Sapp. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[34] K. Chitta, D. Dauner, and A. Geiger. Sledge: Synthesizing driving environments with generative models and rule-based traffic. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024.

[35] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[36] A. Amini, T.-H. Wang, I. Gilitschenski, W. Schwarting, Z. Liu, S. Han, S. Karaman, and D. Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2022.

[37] T.-H. Wang, A. Amini, W. Schwarting, I. Gilitschenski, S. Karaman, and D. Rus. Learning interactive driving policies via data-driven simulation. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2022.

[38] A. Tonderski, C. Lindström, G. Hess, W. Ljungbergh, L. Svensson, and C. Petersson. NeuRAD: Neural rendering for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[39] S. Manivasagam, I. A. Bârsan, J. Wang, Z. Yang, and R. Urtasun. Towards zero domain gap: A comprehensive study of realistic lidar simulation for autonomy testing. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 8272–8282, 2023.

[40] F. Codevilla, A. M. Lopez, V. Koltun, and A. Dosovitskiy. On offline evaluation of vision-based driving models. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[41] M. Bansal, A. Krizhevsky, and A. S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proc. Robotics: Science and Systems (RSS)*, 2019.

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

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

[44] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 2000.

[45] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Proc. Conf. on Robot Learning (CoRL)*, 2021.

[46] K. Renz, K. Chitta, O.-B. Mercea, S. Koepke, Z. Akata, and A. Geiger. Plant: Explainable planning transformers via object-level representations. In *Proc. Conf. on Robot Learning (CoRL)*, 2022.

[47] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. 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.

[48] A. Patil, S. Malla, H. Gang, and Y.-T. 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.

[49] 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.

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

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

[52] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[54] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

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

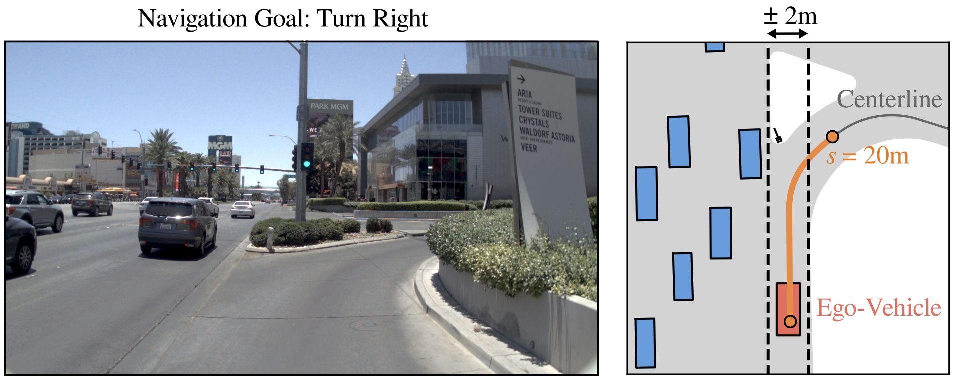

Figure 4: **Navigation Goal.** Given the map, we extract a route centerline (gray) to interpolate a goal point at a distance $s = 20$m (orange) from the rear axle of the ego-vehicle (red). The navigation goal is classified as a right turn, as the lateral distance exceeds 2m to the right.

A 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 [3] for end-to-end driving, there has been inconsistency in prior work [4, 29, 30]. 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 considered 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.

B Implementation Details: NAVSIM

This section provides additional details regarding the metrics and evaluation of NAVSIM.

B.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 is kept fixed in the initial frame of the scene for the entire simulation horizon h of 4s. 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 surrounding agents and plan accordingly. We simulate the trajectory with a controller and motion model (see Sec. B.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 [7, 29, 30], 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.

[1]For details on inconsistencies in nuScenes end-to-end driving evaluation, please refer to PARA-Drive [7].

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 $\pm$2m, as shown in Fig. 4. 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 [28, 29, 30] based on nuScenes [3].

B.2 Predictive Driver Model Score (PDMS)

Similar to established closed-loop benchmarks [8, 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) [8] 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 [3, 48], we find that upsampling 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 2.

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 2: **PDMS.** Subscores with weights and ranges.

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) [42] controller to calculate acceleration and steering values and propagate the ego pose with a kinematic bicycle model [43]. We use the same controller parameters of PDM-Closed and nuPlan [6, 8].

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:

$$\texttt{score}_{\texttt{NC}} = \begin{cases} 1 & \text{no at-fault collision,} \\ 0.5 & \text{one at-fault collision with static class,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

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 [8], 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. 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 [8], 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 versions.

C 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.

C.1 Dataset Support of OpenScene

NAVSIM provides initial support for the OpenScene v1.1 dataset [12], a collection of 120h driving data derived from the nuPlan dataset [8]. 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) [49] and nuPlan's *Dataset License Agreement for Non-Commercial Use* [50]. The licensing covers all available splits in OpenScene. We refer to nuPlan's terms of use [50] 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 orginzational support.

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.

C.2 Dataset Splits & Curation in NAVSIM

Dataset Splits. We summarize all splits used in NAVSIM in Table 3. 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 3). 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

	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`

Table 3: **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).

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

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. **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 4. 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.

D Baselines

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

D.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 [8]. 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) [44] planner from nuPlan [8]. 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 [45], 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 [51], 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 [46], 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 [52] 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 [51], 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.

D.2 Baselines for End-to-End Autonomous Driving

TransFuser. We base our TransFuser [24] 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 [13]. To offer a lightweight sensor baseline, we apply ResNet-34 [52] 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 [53]. 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 [54]. We use a simple FFN on the resulting ego

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

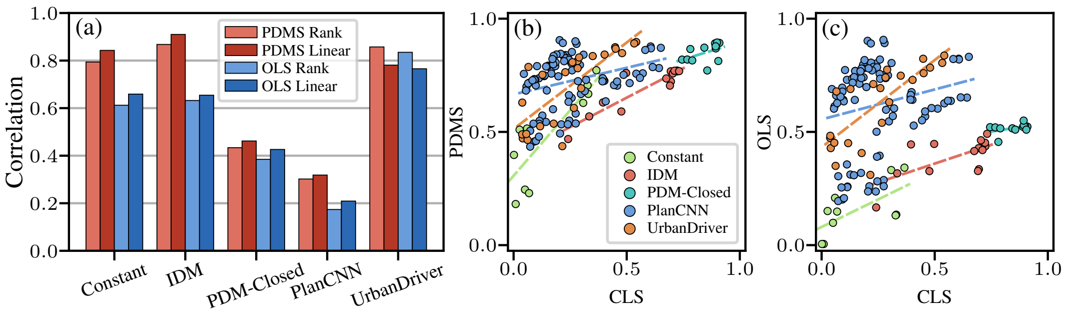

Figure 5: **Planner-Level Alignment of Metrics.** We report the correlation coefficients between open-loop metrics (OLS, PDMS) and the closed-loop score (CLS) for the five planner types considered in our study. The PDMS is better correlated to the CLS for every planner type.

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 [55], a batch size of 96, and a constant learning rate of $1e^{-4}$. We refer to the NAVSIM repository for additional details (see Sec. B).

Latent TransFuser (LTF). For LTF [24], 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 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 [29] 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 [52], while the other model settings remain the same. For the training strategy, we first train a 3D detection model, BEVFormer [47], on `navtrain` for better parameter initialization. This BEVFormer is initialized with a corresponding pre-trained BEVFormer on the nuScenes dataset [3]. 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 [7] 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 [7]. 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.

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

E Additional Results

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

E.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 [8].

Subscores. In Fig. 6, 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.

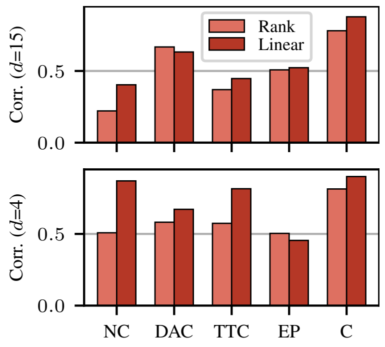

Figure 6: **Correlation of Subscores.**

Filtered Training. In Fig. 7, we show the CLS results for the learned planners from Section D.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.

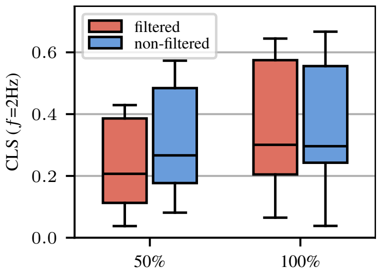

Figure 7: **Training Split CLS.**

Additionally, we present our benchmark results for privileged planning in Fig. 8, 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).

Planner-Level Alignment. The imbalanced distribution of different types of planners in our study may introduce biases into the overall correlations presented in Fig. 3. To address this, we visualize the individual correlations of each planner type in Fig. 5. The correlation values vary depending on metric range and variance of each planner type. Nevertheless, when examining each type individually, the PDMS is better correlated to the CLS than the OLS, and is always positively correlated.

E.2 Revisiting the State of the Art in End-to-End Autonomous Driving

Analyzing TransFuser. In Table 5, we compare several training settings for TransFuser. For the three training seeds in configs A1-A3, we observe a standard deviation of $\pm$ 0.56 in PDMS, which is relatively small compared to variance among training seeds for closed-loop simulations in CARLA [13]. Further, unlike CARLA, NAVSIM is deterministic, and we obtain identical scores when repeating evaluations of a deterministic driving agent. Discarding velocity and acceleration (B1) lowers PDMS by $1.5 - 2.6$, whereas only removing the acceleration (B2) lowers the score by $1.0 - 2.1$. We conclude that while TransFuser benefits from the ego status, it is not purely relying

Config	Parameter	Setting	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
A1	Default config	Seed 1	**98.0**	91.3	**94.2**	100	78.1	83.3
A2		Seed 2	97.7	92.8	92.8	100	79.2	84.0
A3		Seed 3	97.9	**93.0**	93.1	100	**79.3**	**84.4**
B1	Ego status	Goal only	96.8	91.9	91.3	98.6	77.3	81.8
B2		Goal and velocity only	96.7	92.3	91.0	100	77.8	82.3
C1	Camera FOV	60° (1 camera)	96.7	90.2	90.9	100	75.8	80.3
C2		160° (3 cameras)	97.6	91.4	92.7	100	78.1	82.8
C3		240° (5 cameras)	97.8	92.5	93.0	100	79.2	84.1
D1	LiDAR range	F:16, B:16, L:16, R:16	96.9	88.3	91.2	100	74.6	79.1
D2		F:64, B:32, L:32, R:32	97.8	92.7	93.4	100	**79.3**	84.3
D3		F:64, B:64, L:64, R:64	96.8	90.3	91.5	100	76.5	81.0
E1	Supervision	No BEV segmentation	97.4	90.5	92.2	100	77.1	81.6
E2		No 3D detection	97.8	92.7	92.9	100	79.2	84.0

Table 5: **TransFuser Ablations.** The default configuration, which obtains the best results, uses the navigation goal, velocity, and acceleration as ego status inputs. Its camera FOV is around 140° and LiDAR range is 32m to the front (F), back (B), left (L), and right (R). It uses both auxiliary tasks.

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 6: **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.

on the kinematic state for planning. Next, only considering the front camera (C1) with a 60° FOV leads to a small drop in almost all subscores, compared to our default setting of three cropped and concatenated images with a FOV of 140°. However, expanding the FOV with additional cameras does not result in substantially improved scores. Interestingly, restricting the LiDAR range to 16m in all directions (D1), results in a score of 79, which is lower than dropping LiDAR altogether (see LTF in Table 1). Expanding the LiDAR range to 64m in the forward direction (D2) or all directions (D3) does not provide significant improvements. We suspect that changes in the LiDAR range overly simplify or complicate the auxiliary 3D object detection and BEV semantic segmentation tasks, which operate in the LiDAR coordinate frame, hindering effective imitation learning. We check the impact of the auxiliary tasks by excluding them, where performance drops without BEV Segmentation (E1).

Analyzing PARA-Drive. We study the performance influence of auxiliary tasks for PARA-Drive in Table 6. 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.

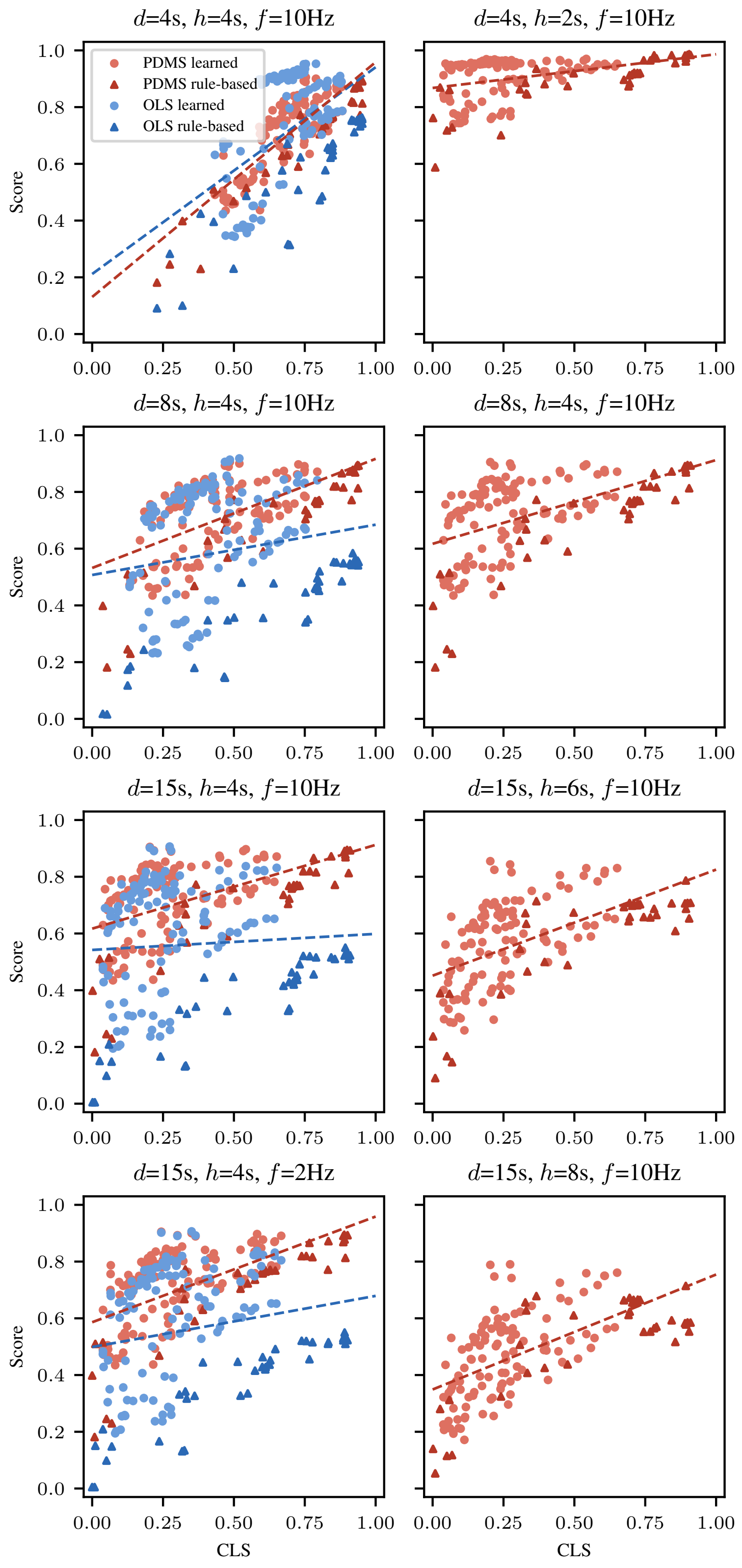

Figure 8: **Closed-Loop Alignment.** We visualize the open-loop metrics (OLS, PDMS) against the closed-loop score (CLS) of the privileged planners. On the left column, we vary the CLS duration d (4s to 15s) and planning frequency f (10Hz vs. 2Hz). The right column shows different PDMS horizons h (2s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