Supplementary Material for Parting with Misconceptions about Learning-based Vehicle Motion Planning

Daniel Dauner[1,2] **Marcel Hallgarten**[1,3] **Andreas Geiger**[1,2] **Kashyap Chitta**[1,2]
[1]University of Tübingen [2]Tübingen AI Center [3]Robert Bosch GmbH
https://github.com/autonomousvision/nuplan_garage

Abstract: In this **supplementary document**, we first provide additional details about the nuPlan dataset and evaluation schemes. Next, we provide information on our standardized benchmark for training and evaluation. Following this, we describe the implementation and training schedules for our baselines. Additionally, we provide detailed descriptions of `PDM`'s components. Finally, we describe our preliminary version of `PDM-Hybrid`, which won the 2023 nuPlan Challenge. The **supplementary video** compares the three evaluation modes in nuPlan and visualizes the operation of rule-based, learned and hybrid planners on different scenarios.

1 nuPlan

1.1 Dataset

The nuPlan dataset compromises 1300 hours of driving data from Las Vegas, Boston, Pittsburgh, and Singapore. The data logs are separated into three sets for training, validation, and testing. The data includes human driving logs, HD map information, and auto-labelled data from an offline perception system. Based on the recordings, short driving scenarios of 15s duration are generated to simulate and measure the capabilities of various planners in open- and closed-loop. The scenarios are categorized into 70 types, of which 14 were used for the 2023 nuPlan challenge. The planner receives a goal position, a sequence of roadblocks to follow, and observations over the past 2s. A planner is required to output an 8s trajectory.

1.2 Simulation

A planner in nuPlan can be simulated in three modes: open-loop, closed-loop non-reactive, and closed-loop reactive.

Open-Loop. The ego vehicle is completely replayed from the human recordings during open-loop simulation. Thus, the planner's output does not influence the vehicle's movement.

Closed-Loop. To simulate the planner in closed-loop, nuPlan uses a two-stage simulation pipeline. First, an LQR controller [1] converts the trajectory into actions. Second, the actions are applied to propagate a kinematic bicycle model [2, 3], approximating the vehicle's motion in coordinate space. The controller and motion model parameters are fixed and cannot be tuned for the challenge. The environment is either non-reactive (replaying the recordings) or reactive by simulating the surrounding vehicles with a lightweight planner.

2 Training & Evaluation

In this section, we describe our training and evaluation benchmark. We will publicly release the configurations and settings required to reproduce our benchmark.

Training. Our proposed training set contains a maximum of 4k samples per scenario, using all 70 scenario types from nuPlan's "training" database files. Overall, this dataset consists of 177,435 training samples.

Evaluation. We evaluate the planners with a maximum of 100 scenarios of the 14 challenge types from nuPlan's validation split, resulting in 1,118 scenarios. To ensure that the scenarios do not overlap, we use a threshold of 15 seconds between the initial frames of subsequent scenarios. At the current stage, we strongly recommend evaluating planners on the validation split and preserving the test split's integrity. We found our benchmark well aligned with the 2023 nuPlan leaderboard.

Hardware. For simulation, training, and runtime analysis, we use an AMD Ryzen 9 7950X processor with 64GB memory and a single NVIDIA RTX 3090.

3 Baselines

In this section, we provide detailed information about the training and implementation of the baseline planners presented in our work.

3.1 Learned Planner

All learning-based planners are trained on our standardized training set, as described in Section 2. The models output an 8s trajectory at 2Hz.

PlanCNN. The `PlanCNN` model is provided in the framework [4]. We use a pre-trained ResNet-50 as CNN encoder. A single linear layer decodes the trajectory. We train PlanCNN with the Adam optimizer, a batch size of 64, and a learning rate of 1e^{-4} for 100 epochs. We divide the learning rate by 10 after 50 and 75 epochs. The CNN is trained with an L1-Loss.

The `Urban Driver` model was introduced in [5] and uses PointNet layers to encode the ego vehicles' motion history, surrounding agents, and the map, represented by polylines. Subsequently, an attention layer aggregates the encoded features before decoding the output trajectory. We use the implementation provided by the nuPlan framework [4] and train the model with an L1-Loss using a batch size of 64 and a learning rate of 1e^{-4} for 60 epochs.

GC-PGP. The `GC-PGP` model was introduced in [6], and extends the state-of-the-art prediction model, called `PGP` [7], for goal-directed ego-forecasting. We refer to Section 5 for a detailed description of the model. Note that a modified variant of `GC-PGP` was used in our preliminary planner. We train the model as proposed in [6] for 70 epochs with a batch size of 32 and a learning rate of 1e^{-4} that is decayed after 40, 50, and 55 epochs by a factor of 0.5.

3.2 IDM

The `IDM` planner utilizes the centerline of lanes as lateral path, and applies the policy from [8] for longitudinal control.

Lateral:. The planner retrieves a sequence of lanes, by applying a Breadth-First-Search (BFS) from the current lane to any lane at the end of the route. If no route is found, the longest lane sequence is returned.

Longitudinal. `IDM` iteratively applies a policy to calculate the longitudinal velocity $\dot{x}_\alpha$ and acceleration $\dot{v}_\alpha$ for the ego vehicle α. By integrating the velocities over time, the planner retrieves the longitudinal ego position x_α, which is interpolated along the centerline to calculate the trajectory samples. Since `IDM` is a parameterized car-following model, each unrolling step requires extracting the states of the leading agent $\alpha - 1$, resulting in the net distance s_α and approaching rate Δv_α:

$$s_\alpha := x_{\alpha-1} - x_\alpha - l_{\alpha-1}, \tag{1}$$

$$\Delta v_\alpha := v_\alpha - v_{\alpha-1}, \tag{2}$$

Parameter	Value	Description
v_0	v_{lane}	Desired velocity. Either the current speed-limit, or $v_{\text{lane}} = 10$ m/s if speed-limit not available.
s_0	1.0 m	Desired net distance to the leading agent $\alpha - 1$.
T	1.5 s	Desired time headway to leading agent $\alpha - 1$.
a	1.0 m/s^2	Maximum acceleration of ego vehicle α
b	3.0 m/s^2	Maximum deceleration (positive) of ego vehicle α
δ	4.0	Acceleration exponent.

Table 1: `IDM` Parameters.

where $l_{\alpha-1}$ is the length of the leading vehicle. Finally, the `IDM` output can be expressed by the following equations:

$$\dot{x}_\alpha = \frac{\mathrm{d}x_\alpha}{\mathrm{d}t} = v_\alpha \tag{3}$$

$$\dot{v}_\alpha = \frac{\mathrm{d}v_\alpha}{\mathrm{d}t} = a\left(1 - \left(\frac{v_\alpha}{v_0}\right)^{\delta} - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right) \tag{4}$$

$$\text{with } s^*(v_\alpha, \Delta v_\alpha) = s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}} \tag{5}$$

where parameters in red are manually selected and summarized in Table 1. Intersections on the route with red traffic lights are considered to be stationary obstacles to obey traffic rules.

4 PDM

4.1 PDM-Closed

Path-Planning. Instead of BFS, `PDM-Closed` utilizes Dijkstra, with the lane length as edge-weights, to search a sequence of lanes for centerline extraction. We found Dijkstra slightly more suitable due to the avoidance of detours while having no substantial effect on run-time.

Observation & Forecasting. `PDM-Closed` generates a forecast of dynamic agents for the planning horizon of 8s with a temporal resolution of 2Hz, and stores the bounding boxes together with static obstacles in occupancy maps. We only consider the nearest 50 vehicles, 10 pedestrians, 10 bicycles, and 50 static objects to the ego agent. Thereby, the planner avoids exploding computation costs when being nearby a large number of entities (e.g. a crowd of pedestrians). Like `IDM`, we add intersections on the route with a red traffic light as stationary objects.

Proposals. We generate proposals by pairing 3 centerline offsets and 5 `IDM` policies at varying target speeds, resulting in 15 proposals. The parameters are summarized in Table 2. Note that we use higher acceleration parameters to favour progress. We iteratively unroll the proposals for 4s at 10Hz, where we update the leading agent states at 5Hz. The shorter proposal horizon and lower frequency of updating the leading agent help in reducing computation costs.

Simulation. We simulate all trajectory proposals for 4s at 10Hz using a faster re-implementation of nuPlan's LQR controller and kinematic bicycle model. Thereby, the proposals are converted into the actual expected trajectory in closed-loop.

Scoring. Our scoring function closely resembles the nuPlan closed-loop metrics [9]. However, we leverage a computationally efficient re-implementation of the metrics to meet the strict runtime requirements of the competition. The scoring considers at-fault collisions, drivable area infractions, and driving direction compliance as multiplicative metrics. Furthermore, the scoring evaluates progress, time-to-collision, and comfortability as weighted metrics. We normalize the progress metric with the highest progress of a proposal free of multiplicative infractions. We use the same weights as nuPlan, but ignore speed-limit compliance and the binary no-progress metric since the

Parameter	Value	Description
o	$\{-1, 0, 1\}$ m	Centerline offsets
v_0	$\{\frac{i}{5} v_{\text{lane}}\}_{i=1,\dots,5}$	Desired velocity. Either the current speed-limit, or $v_{\text{lane}} = 15$ m/s if speed-limit not available.
s_0	1.0 m	Desired net distance to the leading agent $\alpha - 1$.
T	1.5 s	Desired time headway to leading agent $\alpha - 1$.
a	1.5 ms^{-2}	Maximum acceleration of ego vehicle α
b	3.0 ms^{-2}	Maximum deceleration (positive) of ego vehicle α
δ	10.0	Acceleration exponent.

Table 2: `PDM-Closed` Parameters.

`IDM` proposals are naturally bound to comply with the current speed limit, and the no-progress metric cannot be evaluated without privileged knowledge of the human expert's behavior.

Trajectory Selection. Finally, `PDM-Closed` outputs the highest-scoring proposal, which is extended to the entire planning horizon of 8s with the corresponding `IDM` policy. If the best trajectory is expected to collide within 2s, the output is overwritten with an emergency brake maneuver.

4.2 Architectures & Training

We propose two MLP models to investigate open-loop scoring: $\phi_{\texttt{Open}}$ and $\phi_{\texttt{Offset}}$. Both models share architectural attributes and the same training schedule.

Architecture. The $\phi_{\texttt{Open}}$ receives a 120m centerline with 1m resolution ($\mathbf{c}$), and the ego history states ($\mathbf{h}$) consisting of past waypoints and longitudinal, lateral, and angular velocity and acceleration, for the past 2s at 5Hz. Additionally, $\phi_{\texttt{Offset}}$ receives the trajectory ($\mathbf{w}_{\texttt{Closed}}$) of `PDM-Closed` as input, with an 8s duration downsampled to 5Hz. The input features ($\mathbf{c}$, $\mathbf{h}$, and optionally $\mathbf{w}_{\texttt{Closed}}$) are first projected to a 512-dimensional vector with linear layers and then concatenated and forwarded into the MLP. Hereinafter, we apply the same architecture proposed in [10], consisting of two 512-dimensional linear layers and dropout. We use a ReLU activation function for all linear layers. The $\phi_{\texttt{Open}}$ model outputs waypoints $\mathbf{w}_{\texttt{Open}}$ relative to the ego position, whereas $\phi_{\texttt{Offset}}$ predicts offsets to $\mathbf{w}_{\texttt{Closed}}$.

Training. We train $\phi_{\texttt{Open}}$ and $\phi_{\texttt{Offset}}$ on our standardized training split, as described in Section 2. The models with the Adam optimizer, a batch size of 64, and a learning rate of 1e^{-4} for 100 epochs. We divide the learning rate by 10 after 50 and 75 epochs. We use an L1-Loss function for both models. Note that we add $\mathbf{w}_{\texttt{Closed}}$ to the output of $\phi_{\texttt{Offset}}$ during training.

5 2023 nuPlan Challenge

Our preliminary version, which won the 2023 nuPlan competition, was composed of `PDM-Closed` and used a modified version of `GC-PGP` for long-term trajectory correction. The code for `GC-PGP` and our preliminary version will be publicly released.

Goal-conditioned Ego-Forecasting via Graph-based Policy. `GC-PGP` extends the state-of-the-art prediction model, called `PGP` [7], for goal-directed ego-forecasting.

The model receives an ego-centered lane-graph representation, together with observed states of surrounding agents and the ego vehicle. The nodes in the lane-graph compromise polylines of similar length, with directed edges for lanes in proximity or the direction of traffic flow. The lane nodes and dynamics of the surrounding agents and the ego vehicle are encoded with separate Gated Recurrent Units (GRUs). The model aggregates the information by applying Agent-to-Node Attention and Graph Neural Network (GNN) layers, yielding a per-node feature representation.

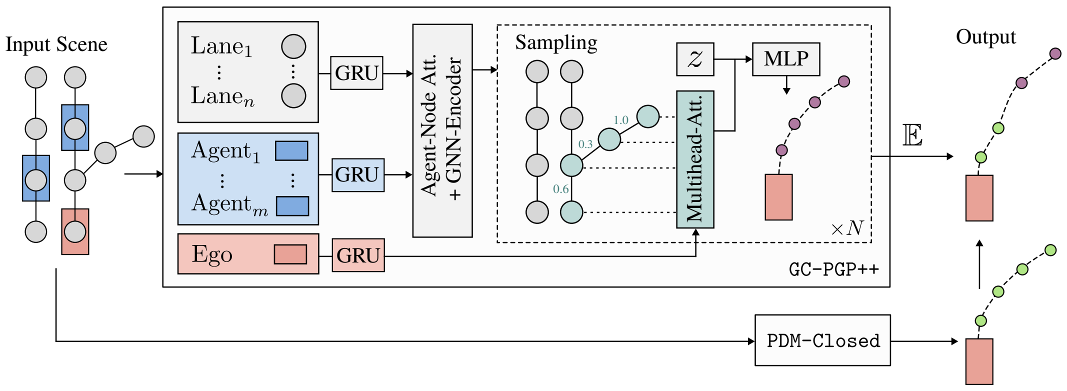

Figure 1: **PDM-Hybrid (Graph).** `GC-PGP++` encodes the input lane-graph and generates trajectory proposals based on lane-traversals. The long-term trajectory of `PDM-Closed` is corrected by the expectation of `GC-PGP++`.

Method	OLS ↑	CLS-R ↑	CLS-NR ↑	Runtime (ms) ↓
`GC-PGP` [6]	82	54	57	100
`GC-PGP++`	**84**	49	50	**84**
`PDM-Hybrid` (Graph)	**84**	**92**	**93**	172
`PDM-Hybrid` (Centerline)	**84**	**92**	**93**	96

Table 3: **Val14 Benchmark**. The preliminary `PDM-Hybrid` (Graph) planner integrates the improved `GC-PGP++` for long-term correction, leading to the same performance as `PDM-Hybrid`.

The node-feature are used to estimate transition probabilities for outgoing edges. Subsequently, traversals across the lane graph are sampled. During inference, `GC-PGP` masks out off-route edges to ensure goal-compliant traversals. Then, a latent-variable model decodes trajectories based on the traversals and the ego-motion encoding. The output trajectories are obtained after a *k*-means clustering. The original version of `GC-PGP` selects the cluster centers with the highest rank as output trajectory.

Modifications. We observe that the goal-conditioned trajectories do not describe disjoint behaviors. Hence we omit the *k*-means clustering and instead calculate the expectation of all decoded trajectories by averaging, as shown in Fig. 1. Our modification, paired with the hard route constraints of `GC-PGP`, led to a slight OLS performance increase while being computationally more efficient. We refer to our modified version as `GC-PGP++`.

References

[1] Y. Tassa, N. Mansard, and E. Todorov. Control-limited differential dynamic programming. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2014.

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

[3] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[4] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. M. Wolff, A. H. Lang, L. Fletcher, O. Beijbom, and S. Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.

[5] 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.

[6] M. Hallgarten, M. Stoll, and A. Zell. From Prediction to Planning With Goal Conditioned Lane Graph Traversals. *arXiv.org*, 2302.07753, 2023.

[7] N. Deo, E. M. Wolff, and O. Beijbom. Multimodal Trajectory Prediction Conditioned on Lane-Graph Traversals. In *Proc. Conf. on Robot Learning (CoRL)*, 2021.

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

[9] Motional. nuplan metrics, 2023. URL https://nuplan-devkit.readthedocs.io/en/latest/metrics_description.html.

[10] 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.