EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Mathematisch-
Naturwissenschaftliche
Fakultät**

Lernbasierte Computer Vision

Masterarbeit

Vehicle Motion Planning using Data-Driven Simulation

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Daniel Wilfried Dauner, `daniel-wilfried.dauner@student.uni-tuebingen.de`, 2023

Bearbeitungszeitraum:	15.02.2023-15.08.2023

Betreuer/Gutachter:	Prof. Dr. Andreas Geiger, Universität Tübingen
Zweitgutachter:	Prof. Dr. Andreas Zell, Universität Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Daniel Wilfried Dauner (Matrikelnummer 4182694), August 15, 2023

Abstract

Autonomous driving systems rely on robust motion planning to safely navigate the vehicle. Prior approaches in motion planning literature for driving are merely evaluated with open-loop metrics or synthetic simulators. The nuPlan dataset offers the first large-scale benchmark in complex real-world scenarios, enabling both open- and closed-loop evaluation. This thesis provides a comprehensive overview of data-driven simulation with insightful experiments on nuPlan. Rule-based approaches, like simple lane-following policies, outperform recent learning-based planners in closed-loop simulation. Surprisingly, a simple multilayer perceptron that ignores all vehicles and only requires a route centerline as scene context achieves the best results in open-loop ego-forecasting. Consequently, we combine our insights and propose a state-of-the-art hybrid planner that won the nuPlan Challenge 2023.

Acknowledgments

First, I want to thank my advisors and team members of the 2023 nuPlan Challenge. Special thanks to Kashyap Chitta for the mentoring on and beyond my thesis. My collaboration with Marcel Hallgarten was nothing short of amazing. I am grateful to Andreas Geiger for the enlightening discussion and for welcoming me into his group.

On a personal note, I owe immeasurable thanks to my parents for supporting me throughout my academic journey. My friends, Dennis and Bálint, have been unwavering sources of motivation and support. To Amy, thank you for your love, patience, and belief in me.

Contents

1 Introduction

Self-driving cars enable automatic navigation of vehicles on public roads without human intervention. Thereby, autonomous vehicles have immense potential to improve and redefine transportation in the future. Specifically for people who are unable to drive, such as disabled or elderly community members, self-driving vehicles could provide independent mobility. Moreover, autonomous vehicles can improve road safety by reducing accidents from human error. Despite significant advances in recent years, the development of reliable, safe, and fully autonomous vehicles remains an unsolved technological challenge.

The predominant approach in the industry for autonomous driving is the modular pipeline architecture. Modular pipelines separate a driving approach into manageable sub-tasks, i.e., perception, prediction, planning, and control. The perception module creates an interpretable understanding of the traffic environment based on high dimensional sensor data (i.e., RGB images or LiDAR scans). Over the last decade, perception systems have transitioned to learning-based techniques powered by advancements in deep learning for computer vision [27, 19, 28, 90, 135]. Publicly available datasets enable researchers to train and evaluate perception methods on real-world driving data [42, 16, 105]. In modular pipelines, the planning module utilizes the perception output to determine the intended future motion of the self-driving vehicle.

The field of motion planning has yet to converge to standardized methodologies, datasets, or evaluation schemes. Research either focuses on rule-based or learned planning, with little overlap in comparing these paradigms. Furthermore, self-driving systems that incorporate planning should be assessed with closed-loop (or online) evaluation, where planning decisions determine the future vehicle states. However, this requires simulation or testing on real vehicles. Real-world testing has substantial safety concerns and high expenses, while simulators exhibit overly simplistic traffic scenarios that are manually designed. Thus, a great amount of planning research is conducted with open-loop (or offline) evaluation on real-world recordings, where planning results are compared to an expert without affecting the vehicle's future. Displacement metrics are most common in open-loop benchmarks [16, 114], as an attempt to measure human-like driving behavior. Nonetheless, prior works demonstrated that open-loop metrics can be misleading for actual driving performance [31, 11].

The nuPlan dataset marks a new era in vehicle motion planning as the first large-scale planning benchmark for data-driven simulation [18]. Specifically, nuPlan offers a platform for open-loop evaluation and closed-loop simulation with easy integration of rule-based and learning-based planners. In contrast to existing simulators [36], nuPlan enables training and simulation based on traffic scenarios encountered in the real world.

Over the course of this thesis, we, a collaborative group of researchers, engaged in the inaugural nuPlan Challenge 2023. The challenge evaluates a submitted planner in several sub-tasks with open-loop and closed-loop metrics. Our proposed method, the Predictive Driver Model (PDM), outperformed a set of 24 international competitors and ranked first on the leaderboard. The main focus of this thesis is to provide detailed descriptions and in-depth experiments of the proposed collection of planners. Since the presented methods emerged in a team effort, I remark team contributions with the plural pronoun “We” and personal contributions with the singular pronoun “I” throughout this thesis. Moreover, I will clearly outline my role in each presented method (see Chapter 4). The contributions of this thesis are:

- A comprehensive introduction to data-driven simulation, with the first empirical study on rule-based and learning-based approaches for vehicle motion planning in nuPlan.
- A family of state-of-the-art planners for open- and closed-loop metrics that expose shortcomings of common evaluation schemes and recent planning approaches.
- A realistic assessment of the current research on vehicle motion planning, with discussions on misconceptions, challenges, and future directions.

The rest of this thesis is structured as follows. In Chapter 2, I present related work in self-driving literature about modular pipeline systems, data-driven simulation, and the task of vehicle motion planning. Specifically, I examine rule-based and learning-based approaches for planning. In Chapter 3, I introduce the nuPlan framework with background information about the dataset, the competition, the evaluation, and the simulation pipeline, together with an overview of baseline planners. Chapter 4 provides a complete description of our proposed methods for planning and ego-forecasting. In Chapter 5, I explain the experimental setup and present the results on the benchmark and leaderboard. Moreover, I conduct further ablation studies to examine the proposed methods. Finally, Chapter 6 discusses the results in consideration of the current research landscape while providing limitations and concluding remarks on this work.

2 Related Work

In this Chapter, I introduce prior work in autonomous driving literature. Section 2.1 presents related literature for modular pipeline systems. Moreover, Section 2.2 deals with data-driven approaches for simulation. In Section 2.3, I introduce prior strategies for planning with rule-based and learning-based systems.

2.1 Modular Pipeline

The modular pipeline approach breaks down the autonomous driving task into separately managed components [63]. Each component is designed for a specific subtask, allowing for independent development and high modularity. Classical modules incorporate perception, prediction, planning, and control.

Perception. The task of perception is to interpret the sensor data and provide a comprehensive understanding of the environment. The perception module typically processes a combination of LiDAR, RGB camera, and localization sensor data. Subfields for perception in autonomous driving include object detection [27, 19, 28, 90, 135], semantic segmentation [69, 10, 70], lane detection [4, 74, 83], occupancy prediction [8, 9, 53, 57], or map construction [78, 93]. In the perception field, several datasets enable training and evaluation with real driving data. These include the KITTI [42], nuScenes [17], or Waymo [105] datasets.

Prediction. The prediction task is to forecast the perceived entities over a defined horizon. In recent literature, prediction primarily focuses on learning-based forecasting of vehicles [33, 34], pedestrians [2, 45], or a combination of actors [61, 100, 133]. However, prior work demonstrated that simple forecasting methods, such as constant velocity or acceleration models, are fast but surprisingly effective [103, 120]. Due to the underlying uncertainty of the task, prediction systems commonly output multiple plausible future trajectories of nearby actors in the environment. There are several datasets and benchmarks for prediction, including Lyft [52], Argoverse 2 [119], or nuScenes [17].

Planning. Given the perception and prediction outcome, the planning module determines the actions and behavior of the ego vehicle's future. The task encompasses diverse objectives, such as safety, comfort, and traffic rule compliance. The module optionally distinguishes between high-level route planning towards a global goal, behavior planning for decision-making, and motion planning for precise future positions of the vehicle [85]. The output of the planning modules is typically a single (or uni-modal) trajectory given to the control module. Previous benchmarks for planning used prediction datasets for open-loop ego-forecasting [16, 52] or simplistic simulators [36]. CommonRoad employs data-driven simulation for planning [3], with a relatively small size of scenario recordings and restricted usage of learning-based planners. The nuPlan benchmark offers the first open- and closed-loop simulator [18], based on a large-scale dataset with integrated rule-based and learning-based planning baselines.

Control. The goal of the control module is to convert the planner output into low-level commands for the ego vehicle. The commands generally include steering, accelerating, and braking. Methods from classical control theory, such as Proportional-Integral-Derivative (PID) controllers, are commonly used in self-driving applications [89, 30, 95]. Optimal control methods, such as Model Predictive Control (MPC) [37], are more advanced by anticipating the system dynamics and formulating the task as an optimization problem.

Ultimately, modular pipelines are the predominant approach in the industry and are used by companies such as Tesla [107], Waymo [115], or Motional [81]. During the 2005 DARPA Grand Challenge, a desert race for driverless cars with increased public attention, several teams with modular systems demonstrated a technological leap for autonomous driving [15, 108]. Since then, there has been an extensive amount of research on modular pipelines [7, 73, 111, 130, 44, 54, 57, 62] with discussion on their benefits. The modules operate independently for the most part, enabling specialized and faster development in parallel. Each module has well-defined interfaces that support generalization and human interpretation. Interpretable representations can (under certain conditions) provide safety guarantees [104].

Another driving paradigm in literature is end-to-end autonomous driving [88, 14, 65, 25, 109, 29, 23, 89, 30], which learn the entire driving task from an input (e.g., raw sensor data) to a driving output (e.g., vehicle controls or trajectories). Thereby, the model can automatically learn features that appear relevant for the driving task. In contrast to modular pipelines, end-to-end circumvent error propagation between intermediate stages. End-to-end driving stacks are generally more straightforward in development and avoid labor-intensive data annotation to train separate modules. On high-fidelity simulators, such as CARLA [36], end-to-end models demonstrated on-par performance compared to a recent modular approach [116].

2.2 Data-Driven Simulation

During the development of an autonomous driving system, rigorous testing and evaluation is critical to ensure safety and robustness. Nonetheless, real-world experiments are expensive, not reproducible, and constitute severe safety concerns. Driving simulators have been vital tools in research [123, 36, 18], enabling fast, cost-efficient, and safe prototyping of novel methods with standardized benchmarks and metrics. However, the simulated environments must be realistic to ensure that techniques and insights transfer to the real world.

Another method is open-loop evaluation which avoids simulation altogether. Open-loop benchmarks have been commonly used in literature for end-to-end models or planning approaches on real-world data [57, 64]. The trajectories of the driving system and human vehicle operator are then compared based on displacement metrics or collision probabilities [17]. However, the driving system does not influence the vehicle or environment during deployment. The open-loop evaluation has no aggregating errors, and a driving system must not be able to recover from potential drifts.

However, there are critical challenges for simulators to ensure realism. Specific sub-challenges include initializing traffic scenes, simulating traffic, or supplying realistic sensor data. Several established driving simulators [123, 36] strongly rely on manually engineered solutions for these tasks. Traffic scenes are initialized based on rules, heuristics, and probabilistic distributions while requiring expertise and manual labor [26]. Traffic simulations often rely on rule-based car-following models, such as the Intelligent Driver Model (IDM) [110], resulting in the simplistic behavior of traffic entities. Furthermore, sensor simulations commonly require 3D computer graphic engines [41]. While these engines offer high flexibility for configuring sensor suites, they need immense computing or supply inadequate realism.

Alternatively, data-driven simulation methods address the challenges by incorporating real-world driving data [3, 18]. Simulators can scale in complexity with growing datasets and allow evaluation with real traffic scenarios.

Scene Initialization. The easiest data-driven method for scene initialization is to sample encountered traffic scenes from the data recordings [18, 75]. Map layouts or traffic dynamics occur as naturally observed on real roads. The simulator can mine specific situations or even optimize the initial parameters for safety-critical scenarios [35, 49, 113]. Another method is to use generative machine learning models to learn the underlying distribution of the real-world data [13, 39]. Such models ideally create plausible traffic scenarios similar to the collected training data.

Traffic Simulation. The task of traffic simulation includes modeling the highly complex and interactive multi-agent environment of real-world roads. For this task, data-driven methods are promising as natural driving behaviors are hard to engineer manually (e.g., lane changes or subtle negotiations). Several methods were proposed for this task [13, 39, 134, 125].

Sensor Simulation. The sensor simulation task involves synthesizing new sensor data, such as images or LiDAR point clouds, from the current viewpoint of the vehicle [5, 127]. For data-driven methods, a learned model has to generate realistic data and account for different movements of the ego vehicle. Therefore, techniques from novel view synthesis, such as Neural Radiance Fields (NeRFs) [79], have been utilized. Additionally, data-driven methods have been used in robotics to enhance the photo-realism of graphic-based simulators [51, 97]. This can avoid distribution shifts when transferring an approach to the real world.

This thesis focuses on the nuPlan dataset for data-driven simulation [18]. Previous planning benchmarks are not built upon large-scale real-world data or do not support closed-loop simulation. The nuPlan simulator supports data-driven scene initialization but only enables log-replay or rule-based (i.e., IDM) traffic simulation. Furthermore, the nuPlan framework does not perform closed-loop sensor simulation (e.g., with novel view synthesis) but provides raw sensor data for about 10% of the dataset.

2.3 Vehicle Motion Planning

This Section examines related work on vehicle motion planning. There are two dominant paradigms for planning in autonomous driving, namely, rule-based and learning-based planning systems.

2.3.1 Rule-based Planning

Rule-based planners employ specific rules to guide the behavior for motion planning (e.g., waiting at a red traffic light). Thereby, rule-based systems can give guarantees for regulations and traffic rules while being highly interpretable compared to learned black-box models. The IDM [110] car-following method is relatively old but remains a relevant technique for rule-based planning or traffic simulation [36, 18]. IDM offers longitudinal control with a mathematical model to follow a leading vehicle at a safe distance based on the current speed, desired speed, or distance to the car in front. Extensions of IDM, e.g. MOBILE [66], enabling lane changes on highway settings. Many planning methods operate in lane-based Frenet frames and employ quintic polynomials for trajectory generation [117, 38].

Rule-based planners mainly rely on perception systems and detailed map information. An alternative approach is offered by predicting affordance indicators that describe the vehicle and environment state tailored for a rule-based decision module [22, 101].

The nuPlan framework integrates rule-based approaches, specifically IDM as baseline planner and background traffic agent in the reactive environment. The method in this thesis builds upon IDM with extensions of several rule-based components.

2.3.2 Learning-based Planning

An emerging field is learned planning, which employs machine learning (ML) models that predict an output trajectory from data. In literature, ML planners strongly vary in learning paradigms, input representations, and evaluation metrics. The following elaborates on behavior cloning (BC) and inverse reinforcement learning (IRL) for expert imitation, together with reinforcement learning (RL).

Behavior Cloning (BC). In BC, a driving policy model is optimized to imitate expert recordings with supervised learning. In 1988, the seminal work ALVINN demonstrated BC with a neural network for road following [88]. More recently, these methods have shown impressive capabilities with end-to-end learning from various inputs, including camera images [126, 25, 84, 12, 29, 121], LiDAR data [96, 40], or combinations of sensor modalities [89, 24, 30]. In modular pipelines, BC can be trained on intermediate representations, such as bird's eye view (BEV) raster [11], or vectorized scene representations [95, 48]. ChaffeurNet [11] encodes rasterized images with convolutional neural networks (CNNs) and incorporates auxiliary losses and perturbations during training to counteract distribution shifts (or covariate shifts) in closed-loop testing. PlanT [95] demonstrates closed-loop planning in CARLA [36], based on trajectory prediction on vectorized scene input with a transformer architecture [112]. Approaches in BC are known to be affected by covariate shifts and causal confusion, where false input-output correlations are exploited during optimization.

Reinforcement Learning (RL). In RL, an agent learns in the environment from a reward function that assigns a value to an action [26]. The reward function is crucial for overall performance but challenging to design manually, specifically for the autonomous driving task. Nevertheless, RL has been effectively applied for self-driving in combination with supervised learning [109, 21], for policy fine-tuning [77, 84], or with privileged input information [68, 132].

Inverse Reinforcement Learning (IRL). Another method for imitation learning is IRL, where the goal is to infer the underlying cost or reward function that motivates the expert's behavior. Classical IRL approaches focus on learning a reward function as a linear combination of handcrafted features [1, 136, 60]. IRL for driving requires a parameterized set of trajectory proposals and learning a cost representation from expert demonstrations with the maximum margin [129, 54, 99] or the maximum entropy method [136, 122, 86]. The trajectory proposal with minimum cost is expected to be similar to the expert's behavior, thus achieving a high reward.

This work primarily focuses on BC in learned planning. Currently, nuPlan does not support RL but offers an imitation learning framework with BC baselines for rasterized and vectorized scene inputs.

3 nuPlan

The nuPlan framework is fully open-source and combines a dataset, a simulator, and an evaluation scheme for the data-driven development of autonomous vehicles. In Section 3.1, I provide a general overview of the collection and annotation of the dataset. Section 3.2 introduces the inaugural nuPlan Challenge 2023 and Section 3.3 explains the simulation pipeline of the framework. Moreover, Section 3.4 introduces the evaluation metrics, and Section 3.5 serves as an overview of the baseline planners in the nuPlan framework.

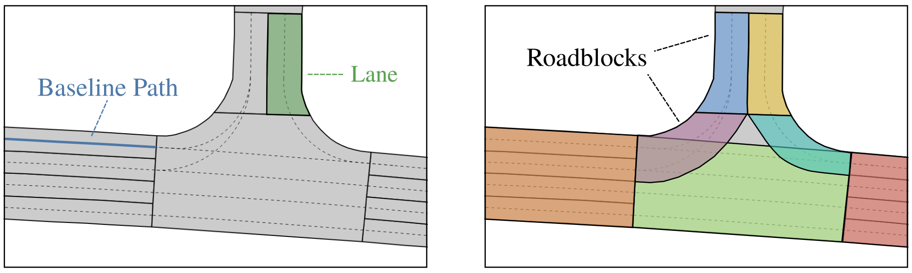
3.1 Dataset

The nuPlan dataset contains driving data of a human vehicle operator from Las Vegas, Boston, Pittsburgh, and Singapore. The driving data is relatively diverse, having both left- and right-hand traffic. The nuPlan framework provides an SQLite[1] database that includes annotations, map information, and vehicle metadata. The database allows extracting training data in various representations and mining short real-world scenarios to simulate and evaluate a planner.

Data Collection. A human vehicle operator collects raw sensor data, providing a ground-truth trajectory for learned planners with imitation learning. The vehicle is equipped with five LiDAR sensors, eight cameras, an Inertial Measurement Unit (IMU), and a Global Navigation Satellite System (GNSS) for localization [80]. The sensor data primarily serves as input for the auto-labeling system. Only 10% of the sensor data is currently available since the unreleased dataset has an immense scale of over 200TB. The database contains about 1300h of driving data in the autolabeled format, where most recordings originate from Las Vegas (838h). Heavy rain conditions and night driving were not considered during data collection.

Data Annotation. The database contains the inferred annotations of an offline perception system, which has two impactful benefits. First, the auto-generated labels do not require extensive storage or costly manual annotation labor, enabling large-scale dataset generation. Secondly, the offline annotation operates without real-time constraints or limited on-car computational resources while considering the whole temporal context. This allows for enhanced detection and globally consistent tracking. The bounding boxes (BB) in nuPlan contain discrete labels for cars, bicycles,

[1] https://www.sqlite.org/



(a) Lane Graph **(b)** Roadblock Graph

Figure 3.1: Graph structure of lanes and roadblocks. The figure shows (a) a highlighted lane and baseline path and (b) the network structure of multiple roadblocks, each in a distinct color.

pedestrians, traffic cones, barriers, construction zone signs, and other generic objects. The offline perception system consists of multi-view fusion [92], PointPillars [71] with CenterPoint [128], and non-causal tracking. Moreover, the perception system detects and infers the traffic-light states for an entire intersection by considering the movement of all actors in the scene.

Maps. The nuPlan framework provides high-definition (HD) maps in 2D BEV that require human effort during annotation. The map includes polygons with semantic classes for roads, intersections, crosswalks, etc. Fig. 3.1 shows the directed graph structure of the map for lanes and roadblocks:

- **Lane:** Map object for a single lane. A lane compromises a polygon, the boundaries, and a baseline path in the lane center. Lanes have directed edges to incoming and outgoing lane objects.
- **Roadblock:** Map object that contains a set of lanes with the same direction. A roadblock contains a polygon, the set of lane objects, and has directed edges to incoming and outgoing roadblocks.

The dataset provides the route of the expert driver as a sequence of roadblocks. Thus, the route information is sparse and does not provide lanes or lane changes taken by the human expert. The source code of nuPlan further categorizes lane and roadblock objects into regular or connector nodes. However, I adopt the terminology from the official documentation[2] and refer to all lanes and roadblocks as regular nodes on the graph structure.

Scenario. Simulation and evaluation in nuPlan is based on scenarios, which are short clips extracted from the dataset with a predefined length (e.g., 15s). Each scenario is categorized with a type descriptor of the current recordings. In total, nuPlan distinguishes between 70 scenario types, e.g., lane change, protected or unprotected

[2] https://nuplan-devkit.readthedocs.io/

	Training		Validation		Test	
	#	%	#	%	#	%
`sg-one-north`	6,663	7.55	660	8.67	626	7.67
`us-ma-boston`	4,888	5.54	484	6.36	357	4.37
`us-nv-las-vegas-strip`	66,402	75.25	5,650	74.22	6,275	76.85
`us-pa-pittsburgh-hazelwood`	10,287	11.66	819	10.76	907	11.11

Table 3.1: Scenario distributions across locations in nuPlan. The table shows the number (#) and percentages (%) of temporally non-overlapping scenarios of the nuPlan splits. Only scenarios with complete goal and route data are considered.

turn, or waiting for crossing pedestrians. The scenario types enable to query diverse, complex, and type-specific scenario sets for the evaluation or training of a planner.

Dataset Splits. The nuPlan database offers separate splits for training, validation, and testing. Notably, the validation and test split are similar in city distribution and overall size, as shown in Table 3.1.

3.2 Competition

This Section provides an overview of the nuPlan Challenge 2023. The leaderboard offers a standardized benchmark with fixed requirements for a planner.

Sub-Challenges. The nuPlan leaderboard evaluates a planner in three sub-challenges: open-loop, closed-loop non-reactive, and closed-loop reactive. In the open-loop challenge, the ego agent follows its recorded log. Thus the planner does not control the vehicle. The open-loop metrics compare the planner output with the expert based on displacements and heading errors. In both closed-loop challenges, the ego-vehicle is simulated in map coordinates with a controller and a vehicle motion model, as described in Section 3.3. The non-reactive challenge replays the environment entirely from the recordings, whereas the reactive challenge simulates non-ego vehicles with an `IDM` planner (described in Section 3.5.1). The closed-loop metrics measure the planning performance based on established driving criteria. In Section 3.4, I comprehensively describe the complete metric structure.

Planner. A planner receives (1) the current and past observation, (2) a sequence of roadblocks as a route, and (3) a goal pose. The observations are available for a 2s history and include the ego states, the detections of the offline perception system with the semantic classes, and the traffic light states. Furthermore, the planner has access to the map and the internal graph structures. The goal pose is typically not reachable in a short simulated scenario. Thus, the roadblock sequence serves primarily for navigation. A planner is required to output an 8s trajectory in the form of the future rear-axle poses of the ego-vehicle. For each simulation iteration, the planner has a time budget of 1s on the evaluation server.

Leaderboard. The planning challenge only considers 14 out of 70 scenario types. The scenarios are part of a hidden dataset that is not publicly available. The scenarios are 15s long and simulated at 10Hz. Each scenario is scored between 0-1, where the metrics depend on the sub-challenge. The sub-challenge score is the average of all scenario scores, and the final score is the average of all sub-challenge scores. A participating team has three submissions to the leaderboard.

3.3 Simulation

In this Section, I explain the closed-loop simulation pipeline of nuPlan. The nuPlan framework sequentially applies a controller to the trajectory of a planner to determine steering and acceleration values, which are fed into a motion model to propagate the ego-vehicle in map coordinate space.

3.3.1 Linear Quadratic Regulator

The nuPlan Challenge applies a Linear Quadratic Regular (LQR) controller to the planner trajectory [72, 106]. This controller originates from optimal control theory to regulate a system by minimizing a cost function. The LQR controller assumes linearity of the system dynamics, i.e., for a time-continuous system given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{3.1}$$

where $\mathbf{x}$ is a state vector, $\mathbf{u}$ is the control value, $\mathbf{A}$ is the state transition matrix, and $\mathbf{B}$ is the control input matrix. Furthermore, the controller has a quadratic cost function J, i.e., given by

$$J = \int_0^\infty (\mathbf{x}^\mathsf{T}\mathbf{Q} + \mathbf{u}^\mathsf{T}\mathbf{R}\mathbf{u})\; dt \tag{3.2}$$

where $\mathbf{Q}$ is a diagonal weighting matrix for the states, and $\mathbf{R}$ is a diagonal weighting matrix for the control inputs. The matrices $\mathbf{Q}$ and $\mathbf{R}$ have manually selected weights to tune the importance of the cost function J for specific states and the desired control inputs, respectively. The optimal solution for this control problem is $\mathbf{u} = -\mathbf{K}\mathbf{x}$, where $\mathbf{K}$ is called the gain matrix and can be found by solving a Riccati equation [72].

The nuPlan framework applies two LQR controllers for longitudinal and lateral control based on the reference time-point of the planner trajectory at a horizon of 1s. The longitudinal controller aims to minimize the velocity error by calculating the optimal acceleration. The lateral controller seeks to reduce the lateral displacement and heading error by adapting the steering rate. When the current velocity is below a threshold, the implementation applies a simple proportional controller for deceleration combined with a steering rate of zero.

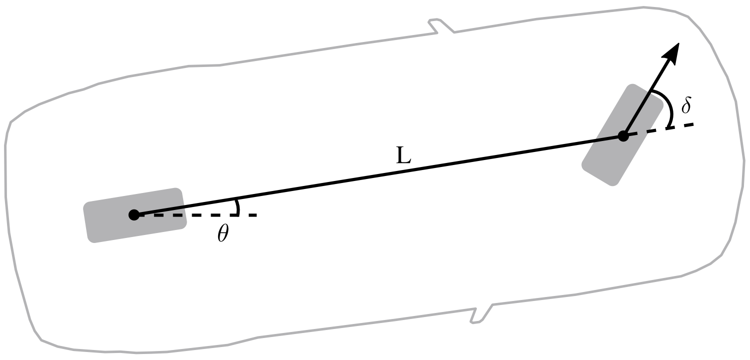


Figure 3.2: Kinematic Bicycle Model. Two axles approximate the motion of a vehicle. Illustrated are the heading angle θ, the wheelbase L, and the steering angle δ.

The framework additionally offers an implementation for the iterative LQR controller [76], which handles non-linearity by repetitive minimization of the quadratic cost function. However, the nuPlan Challenge 2023 employs the LQR controller with a set of fixed parameters (e.g., $\mathbf{Q}$ and $\mathbf{R}$) that cannot be tailored to a planner.

3.3.2 Kinematic Bicycle Model

The kinematic bicycle model is a common method to approximate a car's motion on a 2D coordinate surface [94, 87]. The bicycle model has two wheels that resemble the front and rear axle of the vehicle, as shown in Fig. 3.2. Similar to bicycles and most cars, only the front axle can be used for steering. The model is parameterized at time t by the current rear axle pose (x_t, y_t, θ_t), the longitudinal velocity v_t, the steering angle δ_t of the front wheel, and the constant wheelbase of the vehicle L. The simulation pipeline propagates the bicycle model with the following equations for the duration of Δt

$$x_{t+1} = x_t + v_t \cos(\theta_t)\Delta t \tag{3.3}$$
$$y_{t+1} = y_t + v_t \sin(\theta_t)\Delta t \tag{3.4}$$
$$\theta_{t+1} = \theta_t + \frac{v \tan(\delta_t)}{L}\Delta t. \tag{3.5}$$

Importantly, the nuPlan implementation considers a kinematic bicycle model with a side slip angle β and slip angles α of zero. Before each propagating step, nuPlan updates the velocity v_t and steering angle ψ_t based on the controller values with low-pass filtering. The filter reduces high-frequencies in the acceleration and steering rate of the controller. This ensures a smoother simulation but causes a slower response to control inputs.

Metric	**Weight**	**Range**
Miss Rate (MR)	multiplier	$\{0,1\}$
Average Displacement Error (ADE)	1	$[0,1]$
Average Heading Error (AHE)	2	$[0,1]$
Final Displacement Error (FDE)	1	$[0,1]$
Final Heading Error (FHE)	2	$[0,1]$

Table 3.2: Open-loop Metrics. Summary of metric names, weights, and output values.

3.4 Metrics

Quantifying the broad spectrum of requirements for an autonomous vehicle is a complex task. Some objectives are optional but preferred (e.g., comfort), while other behaviors are indispensable (e.g., collision avoidance).

The nuPlan Challenge has two metric structures for open-loop and closed-loop that follow a similar principle. Both metrics structures assign a score to a scenario by combining a set of multiplier metrics M and weighted average metrics W.

$$\texttt{scenario\_score} = \left(\prod_{m \in M} \texttt{score}_m \right) \times \left(\frac{\sum_{w \in W} \texttt{weight}_w \times \texttt{score}_w}{\sum_{w \in W} \texttt{weight}_w} \right) \tag{3.6}$$

Since all multiplier and weighted scores are in the interval $[0,1]$, the scenario score lies in the same range. Higher scores are better. Thus the multiplier metrics generally carry greater importance. In the following, I will describe the metrics of each sub-task in more detail.

3.4.1 Open-loop

The planner does not influence the motion of the car during open-loop simulation. Therefore, the open-loop metrics only compare the planner output to the trajectory of the human expert driver. The open-loop score consists of four weighted average metrics and one multiplier metric, which I summarize in Table 3.2.

All open-loop metrics operate on the simulation iterations I with a frequency of 1Hz. At each simulation iteration, the framework down-samples the planned trajectories to 1Hz with the future horizon steps $H_t = \{1,\dots,8\}$. All metrics compare the planner and expert over the comparison horizon steps $H_c = \{3,5,8\}$. Note that the indices of H_t and H_c also symbolize the future seconds of the trajectories. For each iteration $i \in I$, I denote the expert trajectory τ_i and the planner trajectory $\hat{\tau}_i$ as $\{(\mathbf{x}_t^i, \theta_t^i)\}_{t \in H_t}$ and $\{(\hat{\mathbf{x}}_t^i, \hat{\theta}_t^i)\}_{t \in H_t}$, respectively.

Miss Rate (MR) within bound. The `MR` is the only multiplier metric in open-loop. The metric considers a trajectory $\hat{\tau}_i$ at iteration $i \in I$ to be a miss if the displacement exceeds threshold $\texttt{tresh}(c)$ at the corresponding comparison horizon $c \in H_c$. The miss rate is given by

$$\texttt{MR} = \frac{1}{|I|}\sum_{i\in I}[\exists c \in H_c : \texttt{thresh}(c) < \|\mathbf{x}_c^i - \hat{\mathbf{x}}_c^i\|_2], \tag{3.7}$$

where $\texttt{tresh}(3) = 6\text{m}$, $\texttt{tresh}(5) = 8\text{m}$, and $\texttt{tresh}(8) = 16\text{m}$. The final score is 0 or 1, depending on whether the MR exceeds 30%.

$$\texttt{score}_{\texttt{MR}} = [\texttt{MR} \leq 0.3] \in \{0,1\}. \tag{3.8}$$

Average Displacement Error (ADE) within bound. The ADE metric calculates the L_2 distances as the average of all scenario iterations, comparison horizons, and samples within a comparison horizon.

$$\texttt{ADE} = \frac{1}{|I|}\sum_{i\in I}\left(\frac{1}{|H_c|}\sum_{c\in H_c}\left(\frac{1}{c}\sum_{t=1}^{c}\|\mathbf{x}_t^i - \hat{\mathbf{x}}_t^i\|_2\right)\right). \tag{3.9}$$

Average Heading Error (AHE) within bound. Similarly, the AHE metric calculates the L_1 distance between the heading angles as an average of all scenario iterations, comparison horizons, and samples within a comparison horizon.

$$\texttt{AHE} = \frac{1}{|I|}\sum_{i\in I}\left(\frac{1}{|H_c|}\sum_{c\in H_c}\left(\frac{1}{c}\sum_{t=1}^{c}\|\theta_t^i - \hat{\theta}_t^i\|_1\right)\right). \tag{3.10}$$

Final Displacement Error (FDE) within bound. The FDE metric calculates the L_2 distances for all $c \in H_c$, and averages over the scenario iterations and comparison horizons.

$$\texttt{FDE} = \frac{1}{|I|}\sum_{i\in I}\left(\frac{1}{|H_c|}\sum_{c\in H_c}\|\mathbf{x}_c^i - \hat{\mathbf{x}}_c^i\|_2\right). \tag{3.11}$$

Final Heading Error (FHE) within bound. Likewise, the FHE metric calculates the L_1 distance between the heading angles for all $c \in H_c$, and averages over the scenario iterations and comparison horizons.

$$\texttt{FHE} = \frac{1}{|I|}\sum_{i\in I}\left(\frac{1}{|H_c|}\sum_{c\in H_c}\|\theta_c^i - \hat{\theta}_c^i\|_1\right). \tag{3.12}$$

Lastly, the scores of a weighted metric $w \in \{\texttt{ADE},\texttt{AHE},\texttt{FDE},\texttt{FHE}\}$ is normalized to the range $[0,1]$, with the following equation

$$\texttt{score}_w = \max\left(0, 1 - \frac{w}{\texttt{max}_w}\right) \in [0,1]. \tag{3.13}$$

for the distance parameters $\texttt{max}_{\texttt{ADE}}$ and $\texttt{max}_{\texttt{FDE}}$ of 8 meters, and heading parameters $\texttt{max}_{\texttt{AHE}}$ and $\texttt{max}_{\texttt{FHE}}$ of 0.8 radian.

Metric	Weight	Range
No at-fault Collisions (NC)	multiplier	$\{0, \frac{1}{2}, 1\}$
Drivable Area Compliance (DAC)	multiplier	$\{0, 1\}$
Driving Direction Compliance (DDC)	multiplier	$\{0, \frac{1}{2}, 1\}$
Making Progress (MP)	multiplier	$\{0, 1\}$
Time to Collision (TTC)	5	$\{0, 1\}$
Ego Progress (EP)	5	$[0, 1]$
Speed-limit Compliance (SC)	4	$[0, 1]$
Comfort (C)	2	$\{0, 1\}$

Table 3.3: Closed-loop Metrics. Summary of metric names, weights, and output values.

3.4.2 Closed-loop

The nuPlan framework creates a closed-loop score for a simulated scenario by combining four multiplier and four weighted average metrics, as shown in Table 3.3.

No at-fault Collisions (NC). A collision occurs when the bounding boxes of a detection track and the ego vehicle intersect. The metric only penalizes at-fault collisions because the detection tracks are primarily non-reactive. At-fault cases are (1) collision with a stationary detection track, (2) ego-front collision with a non-stationary detection track, and (3) ego-side collision when the ego-vehicle is in an intersection or multiple lanes. Conversely, collisions are not at-fault if the ego-vehicle is stationary or in a single lane during a side collision. Given a first bounding box intersection, the metric classifies the collision and ignores the detection track for future frames. Furthermore, the metric distinguishes between at-fault collisions with dynamic detection tracks (vehicles, pedestrians, and bicycles) and static detection tracks (e.g., traffic cone, generic object). The score is given by

$$\texttt{score}_{\texttt{NC}} = \begin{cases} 1, & \text{no at-fault collision} \\ 0.5, & \text{one at-fault collision with static track} \\ 0, & \text{otherwise} \end{cases}. \tag{3.14}$$

Drivable Area Compliance (DAC). The metric is either $\texttt{score}_{\texttt{DAC}} = 0$, if the ego-vehicle leaves the drivable area, or $\texttt{score}_{\texttt{DAC}} = 1$ otherwise. The drivable areas are lanes, intersections, or parking lots. Small violations of up to 0.3m are allowed due to the over-approximation of the ego's bounding box.

Driving Direction Compliance (DDC). The planner is penalized when driving into oncoming traffic. The metrics accumulate the distance $d_{\texttt{DDC}}$ where the ego vehicle's center moves in the opposite direction of a lane. The score is given by

$$\texttt{score}_{\texttt{NC}} = \begin{cases} 1, & \text{if } d_{\texttt{DDC}} \leq 2\text{m} \\ 0.5, & \text{if } 2\text{m} < d_{\texttt{DDC}} \leq 6\text{m} \\ 0, & \text{otherwise} \end{cases} \tag{3.15}$$

Making Progress (MP). This metric is based on the ego progress result $\texttt{score}_{\texttt{EP}}$ and calculated by

$$\texttt{score}_{\texttt{MP}} = [\texttt{score}_{\texttt{EP}} > 0.2] \in \{0,1\}. \tag{3.16}$$

Therefore, the planner is penalized if it has less than 20% of the expert's progress, e.g., when the planner gets stuck in traffic.

Time to Collision (TTC) within bound. The metric projects the ego vehicle and detection tracks at a constant velocity and heading angle with a step size of 0.1s. $\texttt{TTC}$ is the minimum time in any iteration until the ego vehicle collides with a detection track. The metric considers front collisions or collisions when the ego vehicle is in multiple lanes or intersections. Moreover, the metric ignores detection tracks behind the ego vehicle entirely. The result is either $\texttt{score}_{\texttt{TTC}} = 1$, if $\texttt{TTC} > 0.95$s, or $\texttt{score}_{\texttt{TTC}} = 0$ otherwise.

Ego Progress (EP). This metric first extracts the human expert route as a sequence of roadblocks. The expert progress $d_{\texttt{expert}}$ is the distance along the baseline paths of the expert lane sequence. The ego progress $d_{\texttt{ego}}$ is the distance along the baseline paths of the ego lanes if the lanes are in the expert roadblock sequence. The score is given by

$$\texttt{score}_{\texttt{EP}} = \min\left(1, \frac{\max(d_{\texttt{ego}}, 0.1\text{m})}{\max(d_{\texttt{expert}}, 0.1\text{m})}\right) \tag{3.17}$$

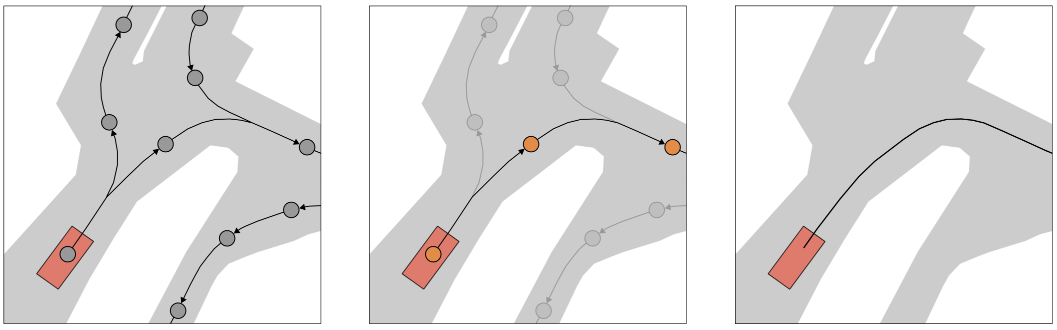
where the planner is not penalized when the expert is stationary (i.e., $d_{\texttt{expert}} < 0.1$). Furthermore, if $d_{\texttt{ego}} < -0.1$m, the metric overwrites $\texttt{score}_{\texttt{EP}} = 0$, in order to penalize the planner for negative progress.

Speed-limit Compliance (SC). The metric extracts the ego-speed $v^i_{\texttt{ego}}$ and speed-limit of the current lane v^i_{lane}, for all scenario iterations $i \in I$. The score calculates the average speed-limit violation and normalizes the result with the equations

$$\texttt{avg\_violation} = \frac{\Delta t}{T} \sum_{i \in I} \max(0, v^i_{\texttt{ego}} - v^i_{\texttt{lane}}) \tag{3.18}$$

$$\texttt{score}_{\texttt{SC}} = \max\left(0, 1 - \frac{\texttt{avg\_violation}\cdot}{\texttt{max\_violation}}\right). \tag{3.19}$$

where $\Delta t = 0.1$s is the iteration interval, and $T = 15$s the scenario duration. The violation score is normalized with $\texttt{max\_violation} = 2.23$ m/s (roughly 5 miles per hour).



(a) Lane Graph. **(b)** Shortest Path. **(c)** Centerline.

Figure 3.3: Centerline Extraction. (a) Given the current lane graph, (b) the `IDM` planner determines the shortest path of lane nodes. (c) The baseline paths of the lane nodes form the centerline.

Comfort (C). This metric verifies that kinematic statistics are within comfortable ranges, which were determined by the challenge organizers based on the expert recordings. Specifically, the metric checks if the longitudinal acceleration is in $[-4.05, 2.40]$ m/s^2, the absolute lateral acceleration is below 4.89 m/s^2, the absolute yaw acceleration is below 1.93 rad/s^2, the absolute yaw velocity is below 0.95 rad/s, the absolute longitudinal jerk is below 4.13 m/s^3, and the jerk magnitude is below 8.37 m/s^3. If the planner complies with all the above conditions, the result is $\texttt{score}_C = 1$, otherwise $\texttt{score}_C = 0$.

3.5 Baselines

The nuPlan framework implements rule- and learning-based baselines. This section describes the baseline planners I use for the experiments.

3.5.1 Intelligent Driver Model

The Intelligent Driver Model (`IDM`) is a common rule-based longitudinal control model designed to follow a leading vehicle while maintaining a safe distance [110]. In nuPlan, the `IDM` planner is a baseline planner and controls the vehicles in the reactive closed-loop challenge. The planner first extracts a lateral path from the provided lane graph and then applies the `IDM` policy for longitudinal control.

Lateral. The planner applies a Breadth-First-Search (BFS) algorithm on the lane graph, as shown in Fig. 3.3. BFS finds the shortest sequence of lanes along the route from the current lane to any lane in the last on-route roadblock. If BFS can not determine a path, the planner considers the furthest lane sequence along the route. The planner connects the baseline paths from the BFS lane sequence to a single line,

Parameter	Value	Description
v_0	v_{lane}	Desired velocity. Either the current speed-limit, or $v_{\text{lane}} = 10\ \text{ms}^{-1}$ 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 ms^{-2}	Maximum acceleration of ego vehicle α
b	3.0 ms^{-2}	Maximum deceleration (positive) of ego vehicle α
δ	4.0	Acceleration exponent.

Table 3.4: Baseline parameters of the `IDM` planner.

which I call the **centerline**. The centerline is a condensed representation of the route with the underlying prior of staying in the center of the road.

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:

$$s_\alpha := x_{\alpha-1} - x_\alpha - l_{\alpha-1}, \tag{3.20}$$
$$\Delta v_\alpha := v_\alpha - v_{\alpha-1}, \tag{3.21}$$

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

$$\dot{x}_\alpha = \frac{\mathrm{d}x_\alpha}{\mathrm{d}t} = v_\alpha \tag{3.22}$$
$$\dot{v}_\alpha = \frac{\mathrm{d}v_\alpha}{\mathrm{d}t} = a\left(1 - \left(\frac{v_\alpha}{v_0}\right)^{\delta} - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right) \tag{3.23}$$
$$\text{with } s^*(v_\alpha, \Delta v_\alpha) = s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}} \tag{3.24}$$

where parameters in red are manually selected and summarized in Table 3.4. Intersections on the route with red traffic lights are considered to be stationary obstacles to obey traffic rules.

3.5.2 PlanCNN

The `PlanCNN` model is a learning-based baseline planner provided in nuPlan [95]. The planner converts the input from the map and offline perception system into a rasterized BEV image presentation. Specifically, the raster input has four channels that encode (1) the non-ego agents and objects, (2) the ego vehicle, (3) the baseline

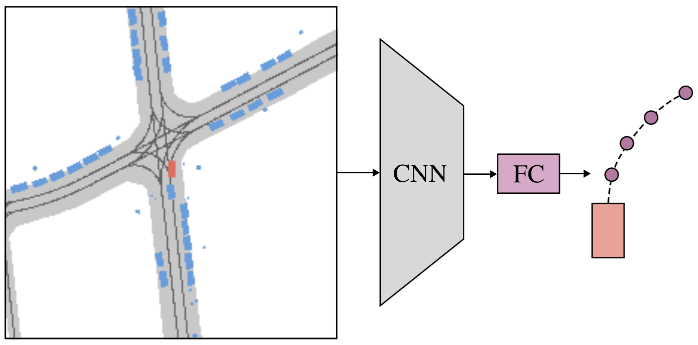


Figure 3.4: **`PlanCNN`.** The input image for the CNN includes non-ego objects (blue), the ego vehicle (red), the baseline paths (dark-gray), and the map objects (light-gray). A fully connected layer decodes the trajectory waypoints.

paths of nearby lanes, and (4) the polygons of lanes, intersections, crosswalks, and stop lines in varying intensities. An example of the input and architecture of `PlanCNN` is shown in Fig. 3.4.

Given the image representation, the framework allows encoding the input with various CNNs from the timm library [118]. The planner replaces the classification layer with a linear layer for regression of the trajectory poses. The network is trained to imitate the expert trajectory and is similar in design to ChaffeurNet [11].

3.5.3 Urban Driver Model

The nuPlan devkit provides an implementation of the `Urban Driver` model [102]. The `Urban Driver` model was created for closed-loop training with policy gradient optimization [50]. However, this requires iterative unrolling and evaluation of the planner during training and a differentiable simulator. Instead, the nuPlan framework offers open-loop training with imitation learning for `Urban Driver`.

The input of `Urban Driver` consists of the ego vehicle, nearby agents, and map elements, as shown in Fig. 3.5. The ego vehicle and agent features encode the current and past poses (with a history of 1.5s). Lanes are represented by the polygons from the lane boundaries and the baseline paths. Additional map elements are also provided as polygons, such as stop lines and crosswalks. The model projects all points of the input entities to a higher dimension, adds a positional embedding (PE), and applies several PointNet layers [91]. As a result, each entity is represented by a 128-dimensional feature descriptor. The vectors of the feature descriptors are further projected and globally aggregated with multihead-attention (MHA) [112]. A final multilayer perceptron (MLP) decodes the waypoints of the output trajectory.

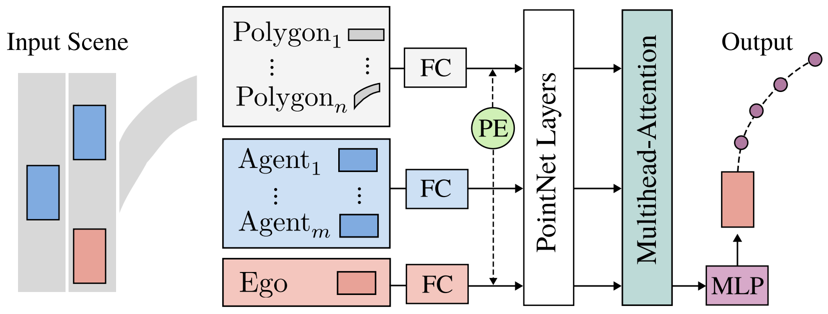


Figure 3.5: **`Urban Driver`.** The model encodes polygons, agents, and ego states of the scene with several PointNet layers. A multihead-attention layer aggregates the entities, and an MLP decodes the trajectory waypoints. (Figure inspired by [102])

3.5.4 Log-Replay Planner

The log-replay planner of nuPlan allows one to compare a planner to the human vehicle operator. Contrary to all presented planners in this thesis, the log-replay planner receives privileged future information about the scenario. Given this information, the log-replay planner outputs the ground-truth trajectory of the human expert at every iteration. Thereby, the planner perfectly imitates the expert in open-loop. However, the planner does not account for controller errors in closed-loop and does not consider the reactive vehicles in the third sub-challenge.

4 Methods

In this chapter, I present the components and fundamental concepts of the proposed planners. As part of this thesis, I participated with my fellow team members in the 2023 nuPlan Challenge. The challenge requires robust planning during closed-loop simulation and the ability to imitate the human driver in open-loop evaluation.

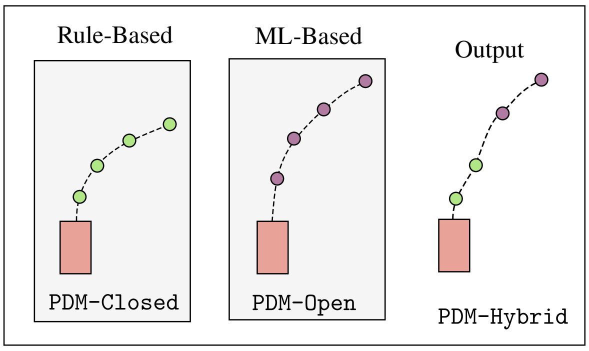


Figure 4.1: The `PDM-Hybrid` planner fuses a short-term trajectory (green) from the rule-based `PDM-Closed` component, and a long-term trajectory (violet) from the learned `PDM-Open` module.

Given these requirements, we propose the Predictive Driver Model (`PDM`), a modular approach that performs planning and long-term correction separately, as shown in Fig. 4.1. Specifically, we combine rule-based planning (`PDM-Closed`) in Section 4.1, and learning-based ego-forecasting (`PDM-Open`) in Section 4.2. Finally, I describe our hybrid planner (`PDM-Hybrid`) in Section 4.3, which our team designed for the nuPlan Challenge.

4.1 Rule-Based Planning

Model Predictive Control (MPC) is a technique that utilizes an internal model to predict control outcomes over a finite horizon. Thereby, MPC can optimize control values for a task-dependent objective while anticipating the future environment.

Our rule-based planner extends the `IDM` baseline with concepts from MPC. Specifically, the planner employs a discrete set of trajectories that are simulated and scored in the forecasted environment. We refer to the planner as `PDM-Closed` since the method contributes to closed-loop performance in our hybrid planner. `PDM-Closed` determines

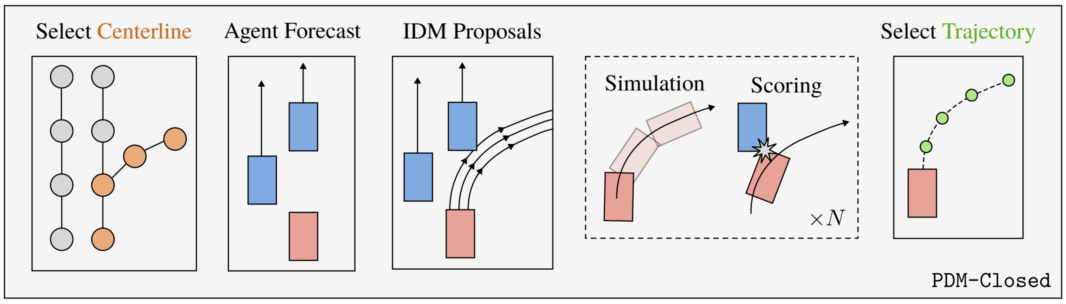


Figure 4.2: `PDM-Closed`**.** The rule-based planner extracts a centerline, projects the agents in the scene, and generates a set of IDM proposals. After each proposal is simulated and scored, the planner selects the output trajectory proposal.

an output trajectory with several components, shown in Fig. 4.2, which I describe in the following.

Centerline. `PDM-Closed` applies a Dijkstra algorithm for centerline extraction, where the length of a connected lane acts as edge-weight. Note that BFS (used by `IDM`) corresponds to Dijkstra's shortest path when all edge weights are equal. I found Dijkstra's algorithm more robust due to avoiding detours with little effect on runtime.

Observation & Forecasting. The offline perception system in nuPlan provides vectors for the orientation and velocity of each semantic bounding box in the scene. `PDM-Closed` extrapolates all objects with constant speed and heading angle to create occupancy maps over the planning horizon H of 8s at 10Hz. Furthermore, we update the dynamic objects at 5Hz and only consider the nearest 50 vehicles, 25 pedestrians, 10 bicycles, and 50 static objects to the ego agent. Thereby, the planner avoids intensive computation costs when being nearby a large number of entities (e.g., a crowd of pedestrians). Like the `IDM` baseline, we add red traffic lights along the route as stationary objects to the occupancy maps.

Proposals. An `IDM` policy approaches a single target speed hyperparameter (v_0 in (3.23)) in free traffic, leading to insufficient variety in longitudinal control. Therefore, `PDM-Closed` employs five `IDM` policies with distinct target speeds, namely, $\{20\%, 40\%, 60\%, 80\%, 100\%\}$ of the current speed-limit (or 15 m/s if the speed-limit is not available). All `IDM` policies have increased acceleration parameters of $a = 1.5$ m/s^2 and $\delta = 10$ for the exponent. The remaining parameters are identical to the baseline `IDM` values in Table 3.4. Furthermore, following a single centerline leads to insufficient lateral variety. Thus, we apply each policy on three lateral centerline offsets ($\pm$1m and 0m), leading to $N = 15$ proposals in total. Each proposal is generated for a prediction horizon P of 4s at 10Hz, where the leading agents are updated at 5Hz. The shorter prediction horizon and lower update frequency of the leading agent reduce computation costs across the planning stages.

Simulation. The `PDM-Closed` planner simulates all proposal trajectories over the prediction horizon P of 4s at 10Hz with a re-implementation of nuPlan's two-stage pipeline (see Section 3.3). Importantly, the revised LQR controller and bicycle model produce the same output while allowing faster batch-wise simulation of the proposals. As a result, the proposals are converted into the expected movement during closed-loop evaluation.

Scoring. The scoring function evaluates the simulated proposals in the forecasted environment with the closed-loop metrics of nuPlan (see Section 3.4.2). We re-implement the metrics for faster batch-wise computation to meet the strict runtime requirements of the competition. The scoring considers at-fault collisions, drivable area infractions, and driving direction compliance as multiplier metrics. With nuPlan's weight parameters, we average the scores for ego progress, time-to-collision, and comfort. The ego progress is measured along the centerline and normalized as a ratio to the most progressing proposal without multiplier infractions. Our scoring function ignores speed-limit compliance and the binary “making progress” metric. The `IDM` proposals naturally comply with the speed-limit and the “making 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 H of 8s with the corresponding `IDM` policy. If the best trajectory has an expected at-fault collision within 2s, the output is overwritten with a maximum braking force maneuver. This emergency brake ensures a stationary ego-vehicle, e.g., during impact with a non-reactive agent.

The implementation of `PDM-Closed` is based on the `IDM` planner (see Section 3.5.1). My primary role for the challenge was the development of `PDM-Closed`, where I wrote the complete code and rigorously optimized the runtime. I acknowledge the crucial input of my peers, such as employing and simulating multiple trajectory proposals.

4.2 Learning-Based Ego-Forecasting

This section presents a family of machine-learning models used interchangeably in our hybrid planner for open-loop performance. In this thesis, I refer to these models as `PDM-Open`. All models are trained with imitation learning. First, I introduce a multi-modal prediction model our team adapted for uni-modal ego-forecasting in the nuPlan Challenge. Secondly, I present a more straightforward approach that performs ego-forecasting based on a single multilayer perceptron (MLP).

4.2.1 Goal-Conditioned Prediction via Graph-based Policy

For the nuPlan Challenge 2023, our team applied Goal-conditioned Ego-Forecasting via Graph-based Policy (`GC-PGP` [48]) as a module for `PDM-Open`. `GC-PGP` extends a state-of-the-art prediction model, called `PGP` [34], that outputs a set of k possible trajectories as ego-forecasts.

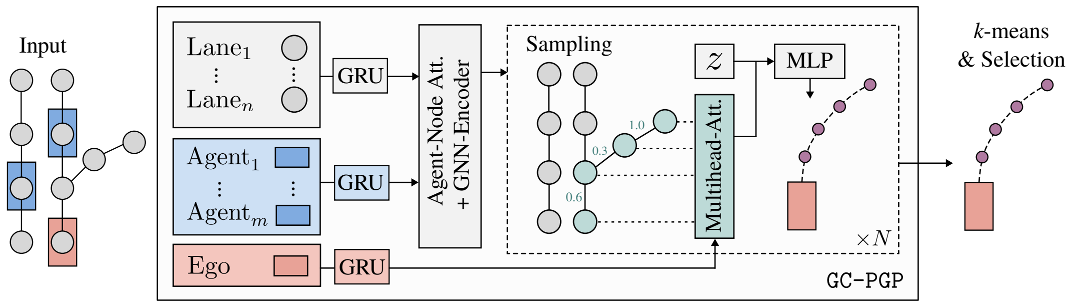


Figure 4.3: `GC-PGP`. The model encodes lane nodes, agents, and ego histories. After applying attention and graph neural network (GNN) layers, the model predicts transition probabilities used to sample lane-traversals. A latent variable model decodes trajectory proposals, which are clustered into k ego-forecasts. Finally, `GC-PGP` outputs the cluster centers with the highest rank.

`PGP`. The model receives an ego-centered lane-graph representation, the ego vehicle state, and observed states of surrounding agents. In contrast to nuPlan's lane graph, `PGP` adds directed edges for nearby lanes and the traffic flow direction. Furthermore, each lane node is restricted to 20m length and input to the network as a polyline (with poses on the baseline path at 1m distance). `PGP` applies three gated recurrent units (GRUs) that encode the past states (1s) of the ego vehicle and agents and the samples of each lane node. The model aggregates the information by applying agent-to-node attention and graph convolutional layers, yielding a per-node feature representation. Based on the node features, an MLP predicts transition probabilities for the outgoing edges of each lane. Given the probabilities, the model samples $N = 1000$ graph traversals. For each traversal, `PGP` aggregates the ego-encoding and the corresponding node features with multihead-attention (MHA). An MLP decodes the proposal trajectories based on the MHA output and a random vector z for longitudinal variety. Finally, `PGP` applies k-means clustering on the proposals and outputs the $k = 10$ cluster centers as the prediction.

`GC-PGP`. The `GC-PGP` model is only trained for ego-forecasting and primarily differs during inference, where the sampled graph traversals are constraint to follow the route provided by nuPlan. Additionally, `GC-PGP` selects the proposal cluster with the highest rank as a uni-modal trajectory output. This `GC-PGP` variant is used in Chapter 5 as an additional baseline planner. As a module for our hybrid planner, we use `GC-PGP`, but we average the trajectory proposal for the final output instead of using clustering.

For the challenge, a team member and first author of `GC-PGP` provided the implementation. Thus, I was not involved in the development. However, I optimized the runtime of `GC-PGP` for the submissions and proposed to average the proposals instead of applying a clustering algorithm.

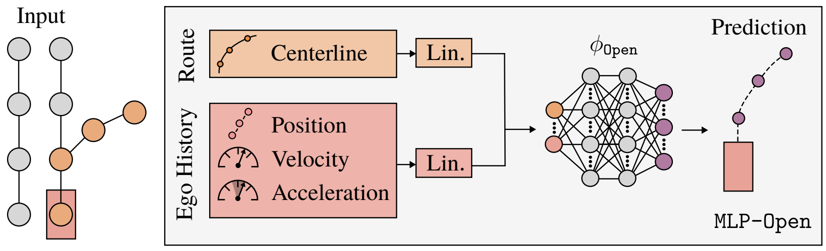


Figure 4.4: **`MLP-Open`.** The model linearly projects the centerline and physical ego-vehicle states, concatenates the outcome, and applies $\phi_{\texttt{Open}}$ with two hidden layers to decode the trajectory.

4.2.2 Centerline-Conditioned Multilayer Perceptron

A recent study showed that a naive MLP achieves competitive open-loop performance on the nuScenes dataset [131] while only taking a direction command and the physical state history of the ego vehicle as input. Inspired by this approach, we propose two ego-forecasting methods, called `MLP-Open` and `MLP-Offset`. Both models consider the ego-state history ($\mathbf{h}$) and a centerline ($\mathbf{c}$) as input, where `MLP-Offset` additionally applies a corrective strategy given the trajectory ($\mathbf{w}_{\texttt{closed}}$) of our rule-based planner from Section 4.2.

`MLP-Open`. The ego-state history covers the past 2s at 5Hz and consists of the position, velocity, and acceleration on the longitudinal, lateral, and angular axis of the vehicle. Furthermore, the model extracts a centerline (identically to `PDM-Closed`) and samples poses over a length of 120m at an interval of 1m. The ego-state history and centerline representation are first scaled to a 512-dimensional vector with a linear layer, concatenated, and input to the MLP $\phi_{\texttt{Open}}$. The architecture of $\phi_{\texttt{Open}}$ incorporates two 512-dimensional linear layers, dropout ($p = 0.1$), and `ReLU` activation functions. A linear output layer regresses the future waypoints, denoted as $\mathbf{w}_{\texttt{Open}} = \phi_{\texttt{Open}}(\mathbf{h}, \mathbf{c})$, for the future 8s at 2Hz.

`MLP-Offset`. Besides the centerline and ego-state history, the MLP $\mathbf{w}_{\texttt{Offset}}$ also receives a linear projection of the waypoints $\mathbf{w}_{\texttt{Closed}}$ from `PDM-Closed` (downsampled to 2Hz). The overall model applies the same architecture to $\phi_{\texttt{Open}}$, but outputs the waypoints

$$\mathbf{w}_{\texttt{Offset}} = \mathbf{w}_{\texttt{Closed}} + \phi_{\texttt{Offset}}(\mathbf{h}, \mathbf{c}, \mathbf{w}_{\texttt{Closed}}), \tag{4.1}$$

where $\phi_{\texttt{Offset}}$ predicts correctional offsets for the trajectory of `PDM-closed`.

For this thesis, I implemented the MLP-based forecasting models in nuPlan and proposed to add the centerline as input. I acknowledge the valuable input of my team members.

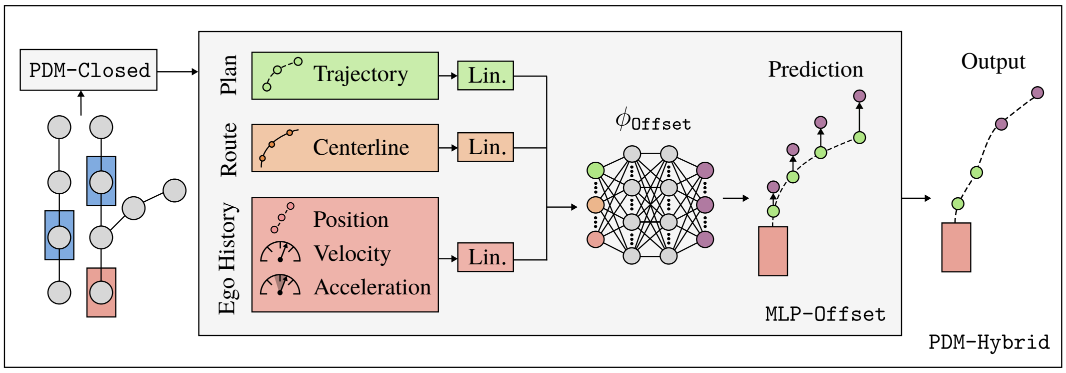


Figure 4.5: **`PDM-Hybrid`.** First, the rule-based `PDM-Closed` module computes a centerline and trajectory (green), which are given to `MLP-Offset` (together with the ego-vehicle states) to predict correctional offsets (violet) for ego-forecasting. The hybrid planner applies the correction only on long-term waypoints to limit the influence of ego-forecasting during closed-loop simulation.

4.3 Hybrid Planning

Given that `PDM-Open` and `PDM-Closed` are separately designed for open-loop and closed-loop performance, our hybrid planner has to combine the strengths of both modules in a single trajectory output. In the following, I describe two options for the hybrid planner to integrate rule-based planning and expert imitation with our modules.

Trajectory Fusion. In practice, the LQR controller used in nuPlan relies exclusively on the first 2 seconds of the trajectory when determining actions in closed-loop simulation. Thus, the planner can consider the `PDM-Closed` trajectory and apply `PDM-Open` beyond a correction horizon C (i.e., 2 seconds). The `PDM-Hybrid` output waypoints (up to the planning horizon H) $\{\mathbf{w}^t_{\texttt{Hybrid}}\}_{t=0}^{H}$ are given by:

$$\mathbf{w}^t_{\texttt{Hybrid}} = \begin{cases} \mathbf{w}^t_{\texttt{Open}} & \text{if } t > C, \\ \mathbf{w}^t_{\texttt{Closed}} & \text{otherwise.} \end{cases} \tag{4.2}$$

where $\mathbf{w}^t_{\texttt{Open}}$ and $\mathbf{w}^t_{\texttt{Closed}}$ are the waypoints at time-step t for `PDM-Open` and `PDM-Closed`, respectively.

Evaluation Detection. Another (arguably less elegant) method is to output the `PDM-Open` or `PDM-Closed` trajectory exclusively, depending on the evaluation mode. This technique serves the competition and has no application in the real world. In the competition, a planner is not directly informed whether it is in an open-loop or closed-loop evaluation. However, `PDM-Hybrid` can detect the mode by simulating an output trajectory of a previous iteration for one step. If there is a notable displacement between the simulated and observed position, the planner assumes to be in open-loop evaluation. This method has two benefits. First, `PDM-Hybrid` does not have to run both modules at each iteration, which saves runtime. Secondly, the `PDM-Open` module is not solely applied beyond a correction horizon, which maintains open-loop performance.

Our final planner `PDM-Hybrid` applies `MLP-Offset` as an ego-forecasting module and uses trajectory fusion to combine the modules, as shown in Fig. 4.5. For the nuPlan Challenge, we used a preliminary planner `PDM-Hybrid`* that employs evaluation detection and integrates our modified `GC-PGP` version for ego-forecasting. However, the evaluation detection is only accessible from the second iteration onwards and requires a significant displacement. Because of that, `PDM-Hybrid`* applies trajectory fusion if a closed-loop evaluation mode cannot be safely ruled out (which includes during the first iteration).

For the competition, I wrote the code structure of the hybrid planner and managed the submissions on the evaluation server. The core concept of combining two trajectories originated from a team member.

5 Experiments

In this Chapter, I conduct several experiments with our proposed methods. In Section 5.1, I describe the benchmark and training scheme for the models and baselines. In Section 5.2, I present the results on the benchmark and challenge leaderboard while briefly describing our competitors. Furthermore, I conduct additional ablations studies and experiments in Section 5.3.

5.1 Benchmark

The challenge leaderboard evaluated the planners on hidden data split and was closed after the competition deadline. Therefore, our team proposes an evaluation benchmark with a fixed training set for fair comparisons.

5.1.1 Evaluation

Similar to the leaderboard, I evaluate the planners for the three sub-challenges. In the following, I describe the validation split, the metrics, and the hardware used for all experiments.

Val14. The evaluation queries 100 scenarios of the 14 challenge scenario types the leaderboard considers. This results in 1,118 scenarios (since not all types have 100 available scenarios). The scenarios are extracted from nuPlan's internal validation split. A threshold of 15s between the initial frames ensures that the scenarios have no temporal overlap.

Metrics. For the planners, I report the score for the three sub-challenges: open-loop score (OLS), closed-loop score non-reactive (CLS-NR), and closed-loop score reactive (CLS-R). The metrics are calculated as described in Section 3.4 but scaled to 0-100 for readability.

Hardware. For simulation, training, and runtime analysis, I use an `AMD Ryzen 7950X` CPU, 64GB memory, and a single `NVIDIA RTX 3090` GPU.

5.1.2 Training

All the models in this study are trained with a maximum of 4,000 samples of all 70 scenario types from nuPlan's internal training database. Overall, the dataset consists

Method	**Rep.**	**OLS ↑**	**CLS-NR ↑**	**CLS-R ↑**	**Time ↓**
`Urban Driver` [102]	Polygon	82	53	50	64
`GC-PGP` [48]	Graph	82	57	54	100
`PlanCNN` [95]	Raster	64	73	72	43
`IDM` [110]	Centerline	38	76	77	27
`MLP-Open`	Centerline	86	50	54	**7**
`MLP-Offset`	Centerline	**87**	61	58	96
`PDM-Closed`	Centerline	42	**93**	**92**	91
`PDM-Hybrid`	Centerline	84	**93**	**92**	96
`PDM-Hybrid`*	Graph	84	**93**	**92**	172
Log Replay	*GT*	*100*	*94*	*80*	-

Table 5.1: Val14 Benchmark. The table compares the open-loop score (OLS), closed-loop score non-reactive/reactive (CLS-NR/CLS-R) and runtime in ms for the planners. Moreover, the table specifies the input representation (Rep.) of each method.

of 177,435 frames. All models output a single 8s trajectory with a waypoint rate of 2Hz and are trained with the Adam optimizer [67].

The `GC-PGP` model is trained according to the original implementation [48]. The first 20 epochs are trained with teacher forcing, where only the ground-truth lane traversal is used for proposal decoding. The lane traversals are randomly sampled from there onward, and training continues for 50 epochs. The trajectory decoder is trained with minADE loss, whereas the policy header (for traversal sampling) is trained with a negative log-likelihood loss. `GC-PGP` uses 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.

The models `PlanCNN`, `Urban Driver`, `MLP-Open`, and `MLP-Offset`, are all trained for 100 epochs with a batch size of 64, an L_1-loss, and a learning rate of $1e^{-4}$ that is decayed after 50 and 75 epochs by a factor of 0.1.

5.2 Results

In this Section, I provide the planning performance on the Val14 benchmark. Furthermore, I present the leaderboard of the nuPlan Challenge 2023, with a description of the competitors.

5.2.1 Benchmark Performance

The results of the Val14 benchmark are summarized in Table 5.1. The `IDM` planner achieves the strongest CLS performance of all learned baseline approaches, with the lowest overall OLS. The results indicate a clear trade-off between both evaluation

Scenario Type	#	**OLS ↑**	**CLS-NR ↑**	**CLS-R ↑**
`behind_long_vehicle`	14	95	100	100
`changing_lane`	70	76	93	95
`following_lane_with_lead`	15	70	95	94
`high_lateral_acceleration`	96	78	87	88
`high_magnitude_speed`	99	86	98	96
`low_magnitude_speed`	100	84	92	90
`near_multiple_vehicles`	85	88	95	89
`starting_left_turn`	100	77	88	89
`starting_right_turn`	98	79	87	88
`starting_straight_traffic_light`	98	85	94	93
`stationary_in_traffic`	98	93	97	95
`stopping_with_lead`	93	95	99	99
`traversing_pickup_dropoff`	99	80	90	90
`waiting_for_pedestrian_to_cross`	53	86	89	89

Table 5.2: **`PDM-Hybrid`.** Results of `PDM-Hybrid` on the 14 scenario types of the Val14 benchmark and across the three sub-challenges. The tables further specifies the number of scenarios (#) per type.

schemes. Learned planners showcase ego-forecasting performance (OLS), whereas rule-based planners exhibit superior planning abilities (CLS). The `PlanCNN` model has the highest CLS results of all learned planners, perhaps because the model discards the ego vehicle state as input. Thereby, `PlanCNN` trades OLS for CLS performance while offering lower runtime. The results contradict the recent trend towards graph- and vector-based representations for planning and prediction [34, 95, 82, 32], showing no advantage in runtime or closed-loop planning performance.

Moreover, the simplest learned model, `MLP-Open`, outperforms all baselines in open-loop while only considering a route centerline as scene context. Interestingly, the `GC-PGP` and `Urban Driver` models receive similar inputs of the route, baseline paths, and the ego-vehicle state. In comparison, the primary strength of `MLP-Open` lies in accurate ego-forecasting for lane-following over a long horizon. I expect that the centerline input simplifies this task, leading to superior OLS, specifically in scenarios that involve curves. I elaborate on this point in Section 5.3.3.

The closed-loop performance of `MLP-Open` is similarly low to baseline models that achieve a high OLS. This indicates a shortcut in ego-forecasting of extrapolating the physical vehicle state, while the map information or other agents have less importance.

The `PDM-Closed` planner substantially improves upon `IDM` in CLS, with little growth in OLS. This underlines the benefits of rule-based planning and the modifications from Section 4.1. Notably, the privileged log replay does not achieve a perfect CLS, partly

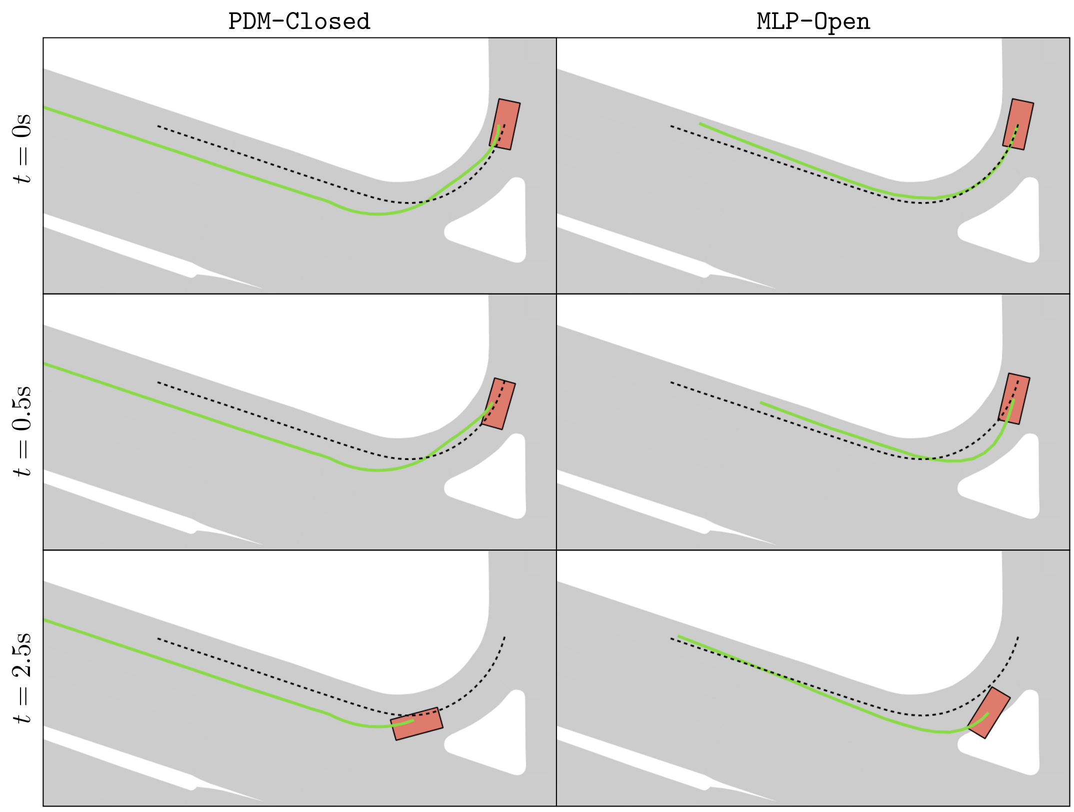


Figure 5.1: Illustration of a nuPlan scenario, with the drivable area (light-gray), the ego-vehicle (red), the planner prediction (green), and the initial 8s human trajectory (dashed-black). At the initial frame ($t = 0$s), `PDM-Closed` has significant displacements from the human trajectory (resulting in low OLS) while safely traversing the curve in subsequent iterations. In contrast, `MLP-Open` accurately predicts the human trajectory but extrapolates the vehicle movement and veers off-road (leading to inferior CLS).

due to nuPlan's LQR controller errors that occasionally cause drifting behaviors from the human trajectory. `PDM-Closed` evaluates all proposals based on the expected simulation outcome. Thus, `PDM-Closed` accounts for controller errors, resulting in similar/superior closed-loop performance compared to log replay.

The `MLP-Offset` model slightly improves upon `MLP-Open` in OLS, indicating some meaningful information around the `PDM-Closed` trajectory for the ego-forecasting task. However, `MLP-Open` cannot retain the CLS numbers provided by `PDM-Closed`.

Finally, `PDM-Hybrid` combines the OLS and CLS strengths of `MLP-Offset` and `PDM-Closed`, respectively. Our preliminary `PDM-Hybrid`* planner achieves identical performance while being less efficient due to added compute of `GC-PGP` as `PDM-Open` module. Notably, the closed-loop abilities of the hybrid planners solely originate from `PDM-Closed`. The learned modules enable long-horizon ego-forecasting and boost the

Method	OLS ↑	CLS-NR ↑	CLS-R ↑	Score ↑
PDM-Hybrid*	83	**93**	**93**	**90**
hoplan [55]	85	89	88	87
pegasus_multi_path [124]	**88**	82	85	85
GameFormer [59]	84	81	84	83
Urban Driver [102]	86	68	70	75
IDM [110]	29	72	75	59

Table 5.3: nuPlan Challenge 2023. The preliminary PDM-Hybrid* planner ranked first on the leaderboard.

open-loop score while adding no clear benefit for the closed-loop planning task.

In Table 5.2, I summarize the PDM-Hybrid performance for each scenario type individually. Given the imbalance and overlap of some scenario types, it is hard to highlight clear tendencies. However, the PDM-Hybrid has difficulties in CLS and OLS for scenarios that involve turns and high lateral acceleration. The CLS-R and CLS-NR results are well aligned, except for the scenario type near_multiple_vehicles where CLS-R is notably lower. The simple IDM background agents regularly block the road, which can hinder progress compared to the recorded human driver.

I compare the PDM-Closed and MLP-Open planner in Fig. 5.1 to illustrate the misalignment between the OLS and CLS metrics. In the initial frame, the PDM-Closed planner significantly differs from the ground truth (leading to low OLS) while steadily following the road with substantial progress (resulting in high CLS). Conversely, MLP-Open achieves high OLS by accurately predicting the human trajectory. Due to extrapolating behavior and accumulating errors, the MLP-Open planners drifts off the driveable area, yielding a poor CLS.

5.2.2 Leaderboard 2023

For the nuPlan Challenge, our team submitted the preliminary PDM-Hybrid* planner to the leaderboard. The leaderboard considers the mean of OLS, CLS-NR, and CLS-R as the overall score (see Table 5.3). PDM-Hybrid* ranked first out of 25 participating teams from 11 countries. I describe the top competing teams in the following.

Horizon Robotics. The hoplan method ranked second [55], and is based on rasterized BEV images as input. The model applies a UNet-like autoencoder on the input image [98], with two heads for (1) the occupancy prediction of dynamic obstacles [56] and (2) a heatmap prediction for the future ego locations [43]. Given a rule-based post-solver [6, 57], an initial trajectory prediction is optimized to avoid occupied areas, to align with the heatmap, and to consider a set of kinematics terms.

Pegasus. The pegasus_multi_path planner ranked third [124] and considered a

Experiment	**OLS ↑**	**CLS-NR ↑**	**CLS-R ↑**	**Time ↓**
w/o forecasting	32	86	86	90
w/o lateral	40	89	89	**58**
w/o longitudinal	**47**	88	88	65
w/o simulation	43	80	80	76
w/o brake	42	91	91	93
`PDM-Closed`	41	**93**	**92**	91

Table 5.4: Module Ablations. The performance across sub-challenges of `PDM-Closed` without (w/o) the listed sub-modules. The runtime is specified in ms.

vectorized representation of the environment combined with a transformer encoder-decoder architecture [112]. The decoder outputs a set of trajectory proposals (and corresponding scores) based on manually tuned anchor seeds [20]. The planner updates the output trajectory every 2 seconds to avoid controller error accumulation. Detected collisions or off-road infractions trigger a trajectory update. The planner selects the highest-scoring proposal without a collision, or applies an emergency brake, if all proposals are expected to collide.

Nanyang Technological University. The `GameFormer` planner ranked fourth and won the innovation award [59]. The planner is based on a centerline that includes lane changes given a set of manually designed criteria. A transformer encoder-decoder architecture, based on [58], predicts the initial trajectory and creates a forecast of nearby agents. The planner projects the ego-trajectory on the centerline and creates centerline occupancies with the agents' forecasts. Finally, a post-solver (based on Gauss-Newton) optimizes the trajectory longitudinally on the centerline for a set of objectives. These include centerline occupancy avoidance, speed-limit compliance, or penalties for uncomfortable driving.

Overall, the leaderboard results are well aligned with the Val14 benchmark (see Table 5.1). Our hybrid planner outperforms all submissions in CLS with a substantial margin while having slightly subpar OLS. All high-ranked planners fundamentally rely on rule-based post-solvers or post-processing steps to counteract the OLS-CLS trade-off. The vectorized scene representation (e.g., in `pegasus_multi_path` or `GameFormer`) shows no performance benefit compared to rasterized inputs (i.e., `hoplan`). Note that the closed-loop behavior of our hybrid planner is exclusively determined by `PDM-Closed`. Consequently, it remains unclear how learned methods or modules can be effectively applied to the closed-loop planning task.

Method	CLS-R ↑	NC ↑	DAC ↑	DDC ↑	MP ↑	TTC ↑	EP ↑	SC ↑	C ↑
w/o forecasting	86	94	99	100	99	88	88	100	85
w/o lateral	89	98	98	100	98	93	86	100	94
w/o longitudinal	88	98	97	100	99	89	89	100	93
w/o simulation	80	91	97	99	100	77	92	100	92
w/o brake	91	97	99	100	99	93	90	100	95
`PDM-Closed`	92	98	99	100	99	94	90	100	95

Table 5.5: Module Ablations CLS-R. Performance of the reactive closed-loop simulation without (w/o) specified sub-modules of `PDM-Closed`.

5.3 Expanded Studies

This Section offers several comprehensive ablation studies that further investigate the `PDM-Closed` planner and `PDM-Open` ego-forecasting model.

5.3.1 Module Ablations

In the following, I ablate several components of `PDM-Closed` to examine their importance across sub-challenges and their impact on runtime. The results are shown in Table 5.4, with a summary of all multiplier and weighted metrics for CLS-R in Table 5.5. Note that runtime measurements have some variability.

Despite its simplicity, the constant velocity forecast is highly effective, adding no significant runtime while improving metrics across simulation modes. In closed-loop, the forecast improves in collision avoidance (NC), near-collision avoidance (TTC), and even comfort (C). Fig. 5.2 shows a nuPlan scenario for `PDM-Closed`, where the planner safely takes an unprotected left turn given the linear forecast of the oncoming traffic. Although not intended for `PDM-Closed`, the forecast yields substantial improvements in OLS since the proposals consider the movement of dynamic agents during generation.

For the proposal ablation, I remove the later offsets (±1m) and separately remove all lateral proposals except one `IDM` policy with 80% of the speed-limit as target velocity. The number of proposals strongly impacts the execution time since they affect the proposal generation, simulation, and scoring. Longitudinal proposals can mainly be processed over a batch dimension (e.g., for leading agent extraction or centerline interpolations), thus, requiring less time compared to lateral proposals. The longitudinal and lateral alations show similar importance for the closed-loop scores. The proposal variety enables the planner to stay in the driveable area (DAC), avoid near collisions (TTC), and achieve progress (EP). Interestingly, removing the

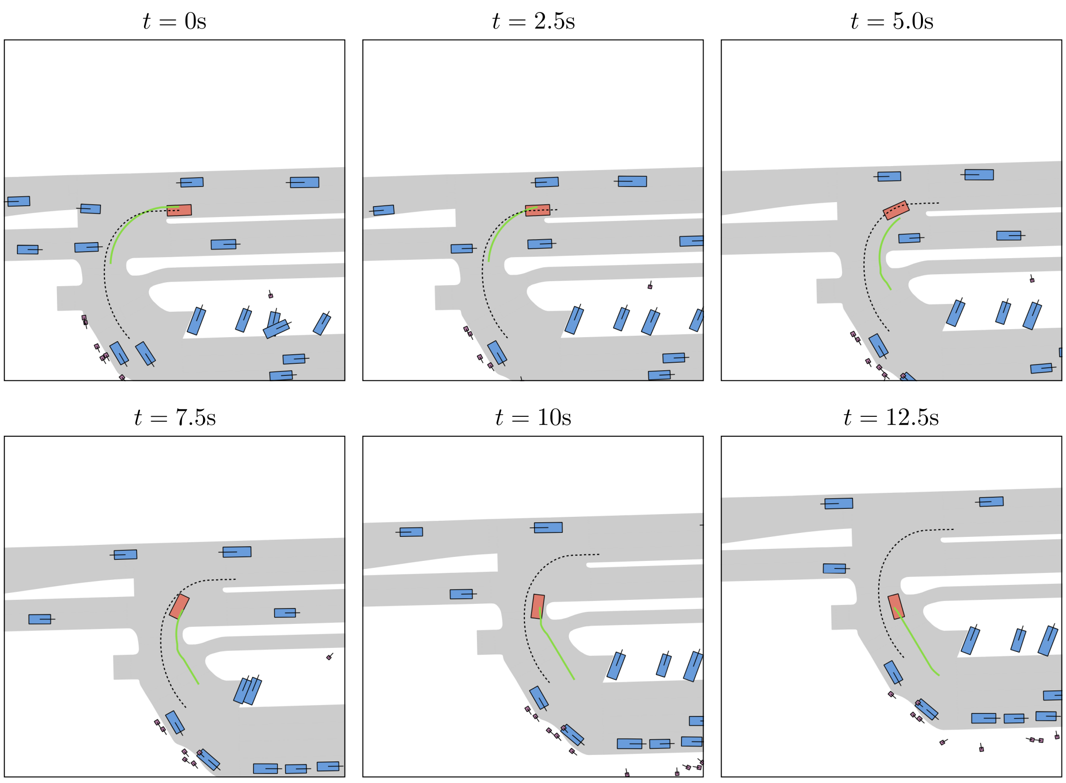


Figure 5.2: `PDM-Closed` in CLS-NR during an unprotected left turn. The ego-vehicle (red) waits for the oncoming vehicles (blue) to pass. Afterward, the planner safely turns into the casino driveway.

longitudinal proposals achieves the highest OLS, further demonstrating the OLS-CLS misalignment.

A fundamental concept of `PDM-Closed` is the proposal scoring based on the simulated vehicle movement. When assuming perfect tracking of trajectory proposals (i.e., removing simulation), the CLS performance of `PDM-Closed` decreases drastically. The scoring function is unable to determine the most appropriate proposal, resulting in more at-fault collisions (NC) and near collisions (TTC).

Lastly, the emergency brake constitutes a simple mechanism for at-fault collision (NC) and near-collision avoidance (TTC).

5.3.2 Cost Function Ablations

This study examines the cost function of `PDM-Closed` used for scoring the proposals. The results across sub-challenges are shown in Table 5.6, with the complete multiplier and weighted metric results of the CLS-R task in Table 5.7.

The re-implemented metrics are thoroughly optimized, allowing batch-wise scoring

Experiment	OLS ↑	CLS-NR ↑	CLS-R ↑	Time ↓
w/o No at-fault Collision	42	91	91	90
w/o Drivable Area Compliance	**43**	82	81	93
w/o Driving Direction Compliance	42	92	92	94
w/o Time to Collision	42	91	90	**87**
w/o Ego Progress	27	77	77	92
w/o Comfort	42	93	92	91
`PDM-Closed`	42	**93**	**92**	91

Table 5.6: Metric Ablations. The performance across sub-challenges of `PDM-Closed` without (w/o) re-implemented metrics for trajectory proposal scoring. The runtime is specified in ms.

of the proposals. Thus, most ablated metrics, except NC and TTC, do not cause a measurable decrease in runtime. Bounding box intersections are slow to examine but are required in both metrics for collision checking.

In closed-loop, the DAC and EP metrics are precious for `PDM-Closed`. DAC filters proposals that lead to off-road infractions, while EP disapproves stationary or low-progressing proposals. Since the planner employs `IDM` policies relative to the centerline, the proposals have little tendency for collisions or driving-direction infractions. Therefore, the NC and DDC metrics have moderate importance but yield improvements in some scenarios (e.g., when encountering controller errors). The TTC re-implementation is well-aligned with nuPlan's version, leading to significant improvement in CLS. Interestingly, `PDM-Closed` achieves the same OLS and CLS performance when dropping the comfort metric. Given its lowest overall weight, the remaining metrics may override the comfort metric.

The multiplier metrics have a filtering role for `PDM-Closed`, assigning a zero score to proposals with severe traffic violations (e.g., collisions). Conversely, the weighted metrics rank the proposals without violations. The continuous progress metric typically plays the primary role in ranking, given that TTC and C are binary 0-1 values. If the progress metric is ignored, the ranking ability vanishes, and the planner exhibits unnecessarily slow driving behavior. Therefore, ignoring the progress metric decreases CLS and OLS, whereas other ablated metrics have no significant effect on open-loop evaluation.

5.3.3 Ego-Forecasting Studies

An interesting insight from the main study in Section 5.2 is that `MLP-Open` outperforms all baselines on the same training and evaluation benchmark, despite the simple architecture and input. In the following, I conduct further experiments to investigate the characteristics of ego-forecasting with `MLP-Open`.

Method	**CLS-R** ↑	**NC** ↑	**DAC** ↑	**DDC** ↑	**MP** ↑	**TTC** ↑	**EP** ↑	**SC** ↑	**C** ↑
w/o NC	91	96	99	100	99	92	91	100	95
w/o DAC	81	98	88	100	99	93	91	100	95
w/o DDC	92	98	99	100	99	94	90	100	95
w/o TTC	90	98	99	100	99	88	91	100	95
w/o EP	77	99	99	100	95	95	59	100	71
w/o C	92	98	99	100	99	94	90	100	95
`PDM-Closed`	92	98	99	100	99	94	90	100	95

Table 5.7: Metric Ablations CLS-R. Performance of the reactive closed-loop simulation without (w/o) re-implemented metrics for proposal scoring of `PDM-Closed`.

Size	**OLS** ↑
0.04 k	79
0.4 k	84
4 k	**86**
40 k	85

(a) Training Size

Cent.	**Hist.**	**CLS-R** ↑	**OLS** ↑
-	-	51	69
-	✓	38	72
✓	-	**54**	85
✓	✓	**54**	**86**

(b) Input Ablations

Method	**OLS** ↑
Baseline	**86**
Shorter centerline	84
Coarser centerline	**86**
Smaller MLP	84

(c) Miscellaneous

Table 5.8: `PDM-Open` Studies. Results of the open-loop score (OLS) and closed-loop score reactive (CLS-R) for several studies. I investigate (a) scaling the training set based on the scenarios per type, (b) ablating the centerline (Cent.) and history (Hist.) input, and (c) varying the centerline parameters or model architecture. The default configuration of `PDM-Open` is highlighted in gray.

First, I test multiple data scales around the original 4k samples per scenario (see Table 5.8a). Specifically, I train `MLP-Open` with a maximum of 40k, 400, and 40 scenarios per type, resulting in about 922k, 20k, and 2k training samples, respectively. Additional training data yields no improvements, resulting in a slightly lower OLS of 85. The model might overfit on majority scenario types that are more dominantly sampled for the large training set. With significantly lower orders of training samples, the `MLP-Open` model still achieves good OLS scores of 84 or 79. I observe that `MLP-Open` learns to extrapolate the vehicle movement along the centerline, which suggests requiring little data for learning.

In Table 5.8b, I compare `MLP-Open` variants when omitting the centerline or history inputs. The model which receives just the current ego state (first row) has an OLS of 69 and thereby still outperforms `PlanCNN` in ego-forecasting (OLS=64, Table 5.1). Adding the history states has little impact on OLS with a drop in CLS. Central for `MLP-Open` is the addition of the centerline input. Prior work [131] that inspired

Method	OLS ↑	MR ↑	ADE ↑	AHE ↑	FDE ↑	FHE ↑
`Urban Driver` [102]	82	94	78	96	60	**94**
`GC-PGP` [48]	82	95	83	94	62	90
`MLP-Offset`	**87**	**97**	**87**	96	**68**	**94**
`MLP-Open`	86	96	86	**97**	67	**94**

Table 5.9: Open-loop Sub-metrics. Individual scores of the weighted and multiplier metrics for open-loop evaluation. Across the sub-metrics, the MLP-based methods outperform the `Urban Driver` and `GC-PGP` baselines.

the design of `MLP-Open` showed competitive performance for 3s ego-forecasting on nuScenes [16], based on ego history states and a high-level command (i.e., turn left, go straight, or turn right). The authors remark that precise ego-forecasting in straight-driving scenarios leads to the open-loop performance of their MLP-based approach. However, the results in nuPlan suggest that more directional information is required for 8s ego-forecasting, e.g., in the form of a centerline.

Moreover, I examine three `MLP-Open` variants in Table 5.8c: a shorter centerline (30m vs. 120m), a coarser centerline (with poses every 10m vs. 1m), and a reduced hidden dimension (512 vs. 256). A smaller hidden dimension and a shorter centerline lead to a decrease in OLS of 84. The coarser centerline demonstrates identical performance to the baseline. Therefore, `MLP-Open` can benefit from distant route information without relying on fine granularity.

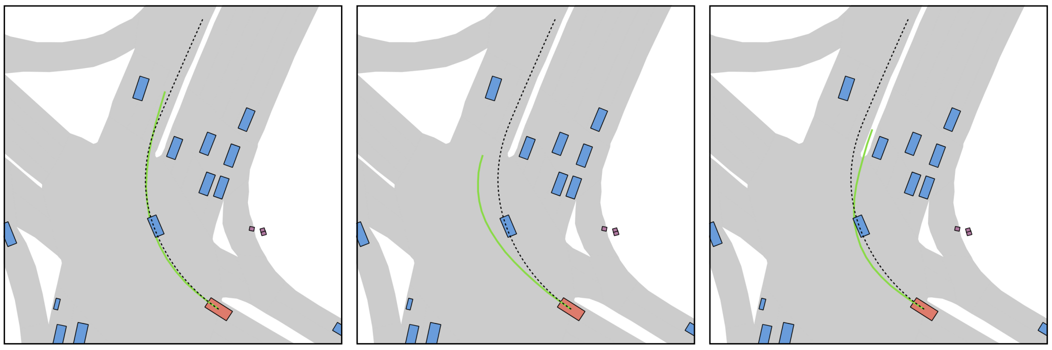

(a) `MLP-Open` (b) `GC-PGP` [48] (c) `Urban Driver` [102]

Figure 5.3: Comparison of a nuPlan scenario (`6bc4abef0de65d50`), with the drivable area (light-gray), the ego-vehicle (red), the planner prediction (green), and human trajectory (dashed-black). The (a) `MLP-Open` model is accurate for ego-forecasting that requires lane-following over a long horizon. In comparison, (b) `GC-PGP` or (c) `Urban Driver` have significantly higher displacements, leading to a zero open-loop score.

For a complete overview, I summarize the OLS sub-metrics for several ego-forecasting models in Table 5.9. The MLP-based models primarily outperform `GC-PGP` and `Urban Driver` in the displacement metrics and the miss-rate. Given the `PDM-Closed` trajectory, the `MLP-Offset` model is indirectly informed about vehicles or traffic lights ahead, which slightly improves the OLS sub-metrics. However, all models achieve similar scores for the average and final heading error, with the exception of `GC-PGP`. Overall, the estimation of the position appears to be more difficult than the orientation.

As mentioned before, the strength of `MLP-Open` primarily lies in forecasting the ego-vehicle along the centerline, as shown in Fig. 5.3. Although `GC-PGP` and `Urban Driver` receive similar inputs of the ego state and the route, `MLP-Open` performs significantly better in OLS for scenarios that involve curves. I expect that accurate lane-following is an effective strategy for ego-forecasting, which is simplified by the centerline-only scene representation.

6 Discussion

In this work, I examine simple yet powerful methods for rule-based planning and learning-based ego-forecasting on the nuPlan benchmark. The results reveal profound misconceptions and shortcomings in the field of vehicle motion planning.

The physical vehicle state and a route path are sufficient input for accurate ego-forecasting. The proposed MLP approach outperforms all methods in open-loop metrics while ignoring actors in the environment and only considering a centerline for map information. Driving without knowledge of the environment is infeasible in the real world, as demonstrated by the poor closed-loop outcomes. Importantly, models with detailed actor and map input (on top of the ego states) show a similar level of performance. This indicates a shortcut of extrapolating the vehicle movement to achieve low displacement errors without learning the underlying expert behavior and decision-making. More research is needed to improve closed-loop planning with imitation-based methods in nuPlan.

The previous insight underlines the fundamental misalignment between open- and closed-loop evaluation. While prior work often relies only on open-loop assessments [57, 64], this thesis demonstrates an inverse relationship to closed-loop metrics that more realistically resemble driving performance. Overall, learned methods have strong ego-forecasting capabilities while driving poorly in closed-loop settings.

Conversely, rule-based planners exhibit the opposite trend with strong closed-loop capabilities. The proposed `PDM-Closed` planner can generalize on the complex nuPlan scenarios while leveraging only a small set of car-following trajectory proposals. `PDM-Closed` can anticipate the closed-loop vehicle movement with the internal simulator, enabling reliable scoring of the proposals. Therefore, I find differentiable simulators [102] or world-models [46, 47] to be promising directions to improve learned planners by incorporating closed-loop dynamics during training.

The `PDM-Hybrid` planner executes the driving maneuvers from the rule-based planner while incorporating a learned model for long-term ego-forecasting. The hybrid design primarily serves the leaderboard structure for the 2023 nuPlan Challenge, where `PDM-Hybrid` claimed overall victory. All top contenders submitted hybrid approaches where rule-based priors ensure closed-loop performance. This indicates that rule-based or hybrid planners exhibit promising potential for future research.

Open-loop metrics are insufficient and misleading for the evaluation of motion planning. In design, displacement metrics favor learned planners while disregarding the evident benefits of simple rule-based methods. Given the findings of my thesis, I discourage the use of open-loop metrics in autonomous driving research, as they potentially hinder advancements toward robust vehicle motion planning.

Limitations. There are important limitations surrounding the planning and simulation approach of this work. All `PDM` planners rely on privileged offline perception and HD map inputs, which may not be available in the real world. Although prior work successfully demonstrated real-world employment of learned planning methods developed based on perfect perception [11, 102, 86], it remains unclear how the insights (i.e., around rule-based methods) transfer to actual modular pipelines. Moreover, the data-driven simulation of nuPlan has inherent limitations. The environment recordings are centered around the human operator. Obstacles can unexpectedly occur ahead of the ego-vehicle in simulation. Since long scenarios would require identical progress of the ego-vehicle and human driver, only short simulations are possible. Therefore, the `PDM` planner is marginally penalized for not executing lane changes. Collisions that occur in between lanes are more strictly declared to be "at-fault", resulting in additional risk attached to lane changes. For CLS-NR, the background vehicles move as observed in the real world while behaving overly aggressively when disregarding the ego-vehicle. In CLS-R simulation, the non-ego vehicles are too passive by strictly following the centerline with an `IDM` longitudinal control. I regard data-driven traffic simulation or initialization as promising directions to refine the reactive environment or to enable long simulations.

Conclusion. The field of motion planning still requires substantial innovation for robust and scalable solutions. Data-driven simulators like nuPlan provide accessible platforms to develop and test novel techniques with real-world scenarios. In this work, I conduct the first rigorous experimental study on nuPlan, where I reveal and investigate several shortcomings of evaluation schemes and learned methods for motion planning. Based on these insights, the proposed **`PDM-Hybrid`** planner combines the strengths of rule-based planning and long-horizon ego-forecasting. Together with my team members, I participated in the 2023 nuPlan Challenge, where the **`PDM-Hybrid`** planner outperformed all competitors and claimed victory.

Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. of the International Conf. on Machine learning (ICML)*, page 1, 2004.

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[3] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[4] Mohamed Aly. Real time detection of lane markers in urban streets. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2008.

[5] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2419–2426. IEEE, 2022.

[6] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2019.

[7] Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, et al. Odin: Team victortango's entry in the darpa urban challenge. *Journal of Field Robotics (JFR)*, 25(8), 2008.

[8] H. Badino, U. Franke, and R. Mester. Free space computation using stochastic occupancy grids and dynamic programming. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV) Workshops*, 2007.

[9] Hernan Badino, Uwe Franke, and David Pfeiffer. The stixel world - a compact medium level representation of the 3d-world. In *Proc. of the DAGM Symposium on Pattern Recognition (DAGM)*, 2009.

[10] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep

convolutional encoder-decoder architecture for image segmentation. *arXiv.org*, 1511.00561, 2015.

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

[12] Aseem Behl, Kashyap Chitta, Aditya Prakash, Eshed Ohn-Bar, and Andreas Geiger. Label efficient visual abstractions for autonomous driving. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[13] Luca Bergamini, Yawei Ye, Oliver Scheel, Long Chen, Chih Hu, Luca Del Pero, Blazej Osinski, Hugo Grimmett, and Peter Ondruska. Simnet: Learning reactive self-driving simulations from real-world observations. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2021.

[14] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *arXiv.org*, 1604.07316, 2016.

[15] M. Buehler, K. Iagnemma, and S. Singh. *The 2005 darpa grand challenge: The great robot race*, volume 36. Springer, 2007.

[16] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv.org*, 2019.

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

[18] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric M. Wolff, Alex H. Lang, Luke Fletcher, Oscar Beijbom, and Sammy 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.

[19] Zhaowei Cai, Quanfu Fan, Rogério Schmidt Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

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

[21] Raphaël Chekroun, Marin Toromanoff, Sascha Hornauer, and Fabien Moutarde.

GRI: general reinforced imitation and its application to vision-based autonomous driving. *arXiv.org*, 2111.08575, 2021.

[22] Chenyi Chen, Ari Seff, Alain L. Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 2722–2730, 2015.

[23] Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. Learning to drive from a world on rails. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021.

[24] Dian Chen and Philipp Krähenbühl. Learning from all vehicles. In *CVPR*, 2022.

[25] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Proc. Conf. on Robot Learning (CoRL)*, 2019.

[26] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *arXiv preprint arXiv:2306.16927*, 2023.

[27] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

[30] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2022.

[31] Felipe Codevilla, Antonio M. Lopez, Vladlen Koltun, and Alexey Dosovitskiy. On offline evaluation of vision-based driving models. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[32] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. Gorela: Go relative for viewpoint-invariant motion forecasting. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2023.

[33] Zhaopeng Cui, Lionel Heng, Ye Chuan Yeo, Andreas Geiger, Marc Pollefeys, and Torsten Sattler. Real-time dense mapping for self-driving vehicles using

fisheye cameras. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2019.

[34] Nachiket Deo, Eric M. Wolff, and Oscar Beijbom. Multimodal Trajectory Prediction Conditioned on Lane-Graph Traversals. In *Proc. Conf. on Robot Learning (CoRL)*, 2021.

[35] Wenhao Ding, Minjun Xu, and Ding Zhao. Learning to collide: An adaptive safety-critical scenarios generating method. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2020.

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

[37] Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on control systems technology*, 15(3):566–580, 2007.

[38] Haoyang Fan, Fan Zhu, Changchun Liu, Liangliang Zhang, Li Zhuang, Dong Li, Weicheng Zhu, Jiangtao Hu, Hongye Li, and Qi Kong. Baidu apollo EM motion planner. *arXiv.org*, 1807.08048, 2018.

[39] Lan Feng, Quanyi Li, Zhenghao Peng, Shuhan Tan, and Bolei Zhou. Trafficgen: Learning to generate diverse and realistic traffic scenarios. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2023.

[40] Angelos Filos, Panagiotis Tigas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *Proc. of the International Conf. on Machine learning (ICML)*, 2020.

[41] Epic Games. Unreal engine. Online: accessed 7-Aug-2023.

[42] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[43] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. In *Proc. IEEE Conf. on Intelligent Transportation Systems (ITSC)*, pages 500–507. IEEE, 2021.

[44] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Matthew A Wright, Joseph E Gonzalez, and Ion Stoica. Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2021.

[45] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks.

In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2255–2264, 2018.

[46] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[47] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proc. of the International Conf. on Machine learning (ICML)*, pages 2555–2565. PMLR, 2019.

[48] Marcel Hallgarten, Martin Stoll, and Andreas Zell. From Prediction to Planning With Goal Conditioned Lane Graph Traversals. *arXiv.org*, 2302.07753, 2023.

[49] Niklas Hanselmann, Katrin Renz, Kashyap Chitta, Apratim Bhattacharyya, and Andreas Geiger. King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[50] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015.

[51] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. Retinagan: An object-aware approach to sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10920–10926. IEEE, 2021.

[52] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Long Chen, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One thousand and one hours: Self-driving motion prediction dataset. In *Proc. Conf. on Robot Learning (CoRL)*, pages 409–418. PMLR, 2021.

[53] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeff Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. FIERY: future instance prediction in bird's-eye view from surround monocular cameras. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021.

[54] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[55] Yihan Hu, Kun Li, Pingyuan Liang, Jingyu Qian, Zhening Yang, Haichao Zhang, Wenxin Shao, Zhuangzhuang Ding, Wei Xu, and Qiang Liu. Imitation with spatial-temporal heatmap: 2nd place solution for nuplan challenge. *arXiv.org*, 2023.

[56] Yihan Hu, Wenxin Shao, Bo Jiang, Jiajie Chen, Siqi Chai, Zhening Yang, Jingyu Qian, Helong Zhou, and Qiang Liu. Hope: Hierarchical spatial-temporal network for occupancy flow prediction. *arXiv.org*, 2022.

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

[58] Zhiyu Huang, Haochen Liu, and Chen Lv. Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. *arXiv.org*, 2303.05760, 2023.

[59] Zhiyu Huang, Haochen Liu, Xiaoyu Mo, and Chen Lyu. Gameformer planner: A learning-enabled interactive prediction and planning framework for autonomous vehicles.

[60] Zhiyu Huang, Jingda Wu, and Chen Lv. Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning. *IEEE Trans. on Intelligent Transportation Systems (TITS)*, 23(8):10239–10251, 2021.

[61] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[62] Bernhard Jaeger, Kashyap Chitta, and Andreas Geiger. Hidden biases of end-to-end driving models. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023.

[63] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*, volume 12. Foundations and Trends in Computer Graphics and Vision, 2020.

[64] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023.

[65] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John Mark Allen, Vinh Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. *arXiv.org*, abs/1807.00412, 2018.

[66] Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model mobile for car-following models. *Transportation Research Record*, 1999(1), 2007.

[67] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2015.

[68] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

[69] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and JamesM. Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014.

[70] Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[71] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[72] Norman Lehtomaki, NJAM Sandell, and Michael Athans. Robustness results in linear-quadratic gaussian based multivariable control designs. *IEEE Trans. on Automatic Control (TAC)*, 26(1):75–93, 1981.

[73] John J. Leonard, Jonathan P. How, Seth J. Teller, Mitch Berger, Stefan Campbell, Gaston A. Fiore, Luke Fletcher, Emilio Frazzoli, Albert S. Huang, Sertac Karaman, Olivier Koch, Yoshiaki Kuwata, David Moore, Edwin Olson, Steve Peters, Justin Teo, Robert Truax, Matthew R. Walter, David Barrett, Alexander Epstein, Keoni Maheloni, Katy Moyer, Troy Jones, Ryan Buckley, Matthew E. Antone, Robert Galejs, Siddhartha Krishnamurthy, and Jonathan Williams. A perception-driven autonomous urban vehicle. *Journal of Field Robotics (JFR)*, 25(10):727–774, 2008.

[74] Jun Li, Xue Mei, Danil V. Prokhorov, and Dacheng Tao. Deep neural network for structural prediction and lane detection in traffic scene. *IEEE Trans. on Neural Networks and Learning Systems*, 28(3):690–703, 2017.

[75] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 45(3):3461–3475, 2022.

[76] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *First International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 222–229. SciTePress, 2004.

[77] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. CIRL: controllable imitative reinforcement learning for vision-based self-driving. *arXiv.org*, abs/1807.03776, 2018.

[78] Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. Maptr: Structured modeling and learning for online vectorized hd map construction. *arXiv preprint arXiv:2208.14437*, 2022.

[79] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[80] Motional. nuplan webpage. https://www.nuplan.org. Online: accessed 8-Aug-2023.

[81] Motional. Voluntary safety self-assessment (vssa). https://motional.com/sites/default/files/inline-files/Motional_Voluntary_Safety_Self-Assessment.pdf, 2021. Online: accessed 7-Aug-2023.

[82] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple and efficient attention networks. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2023.

[83] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018.

[84] Eshed Ohn-Bar, Aditya Prakash, Aseem Behl, Kashyap Chitta, and Andreas Geiger. Learning situational driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[85] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 2016.

[86] Tung Phan-Minh, Forbes Howington, Ting-Sheng Chu, Sang Uk Lee, Momchil S Tomov, Nanxiang Li, Caglayan Dicle, Samuel Findler, Francisco Suarez-Ruiz, Robert Beaudoin, et al. Driving in real life with inverse reinforcement learning. *arXiv.org*, 2022.

[87] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud 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.

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

[89] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion

transformer for end-to-end autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[90] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[91] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[92] Charles R. Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[93] Limeng Qiao, Yongchao Zheng, Peng Zhang, Wenjie Ding, Xi Qiu, Xing Wei, and Chi Zhang. Machmap: End-to-end vectorized solution for compact hd-map construction. 2023.

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

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

[96] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2020.

[97] Stephan R Richter, Hassan Abu AlHaija, and Vladlen Koltun. Enhancing photorealism enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1700–1715, 2022.

[98] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[99] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[100] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 683–700. Springer, 2020.

[101] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. In *Proc. Conf. on Robot Learning (CoRL)*, 2018.

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

[103] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 5(2):1696–1703, 2020.

[104] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv.org*, 2017.

[105] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

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

[107] Tesla. Tesla ai day 2022. https://www.youtube.com/live/ODSJsviD_SU?feature=share&t=3480, 2022. Online: accessed 7-Aug-2023.

[108] Sebastian Thrun, Michael Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia M. Oakley, Mark Palatucci, Vaughan R. Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary R. Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara V. Nefian, and Pamela Mahoney. Stanley: The robot that won the DARPA grand challenge. *Journal of Field Robotics (JFR)*, 23(9):661–692, 2006.

[109] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

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

[111] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al.

Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics (JFR)*, 25(8), 2008.

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

[113] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[114] Waymo. Be an early rider. https://waymo.com/apply, 2019. Online: accessed 18-October-2019.

[115] Waymo. Safety report. https://waymo.com/safety/safety-report, 2021. Online: accessed 7-Aug-2023.

[116] Xinshuo Weng, Peter Karkus, Yulong Cao, Boris Ivanovic, Yue Wang, Yuxiao Chen, Apoorva Sharma, and Marco Pavone. Sms: A safety-enhanced modular stack for autonomous driving. 2023.

[117] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenét frame. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2010.

[118] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[119] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. 2023.

[120] Haoran Wu, Tran Phong, Cunjun Yu, Panpan Cai, Sifa Zheng, and David Hsu. What truly matters in trajectory prediction for autonomous driving? *arXiv.org*, 2023.

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

[122] Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. Watch this: Scalable cost-function learning for path planning in urban environments. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, pages 2089–2095. IEEE, 2016.

[123] Bernhard Wymann, Christos Dimitrakakisy, Andrew Sumnery, Eric Espié, and Christophe Guionneauz. Torcs: The open racing car simulator, 2015.

[124] Weitao Xi, Liangchao Shi, and Guangzhi Cao. An imitation learning method with data augmentation and post processing for planning in autonomous driving.

[125] Danfei Xu, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Bits: Bi-level imitation for traffic simulation. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2023.

[126] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3530–3538, 2017.

[127] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023.

[128] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 11784–11793, 2021.

[129] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[130] Wenyuan Zeng, Shenlong Wang, Renjie Liao, Yun Chen, Bin Yang, and Raquel Urtasun. Dsdnet: Deep structured self-driving network. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 156–172. Springer, 2020.

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

[132] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021.

[133] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Proc. Conf. on Robot Learning (CoRL)*, pages 895–904. PMLR, 2021.

[134] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for

controllable traffic simulation. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2023.

[135] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[136] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2008.