Mathematisch-
Naturwissenschaftliche
Fakultät

Lernbasierte Computer Vision

Research Project

PDM-Lite: A Rule-Based Planner for CARLA Leaderboard 2.0

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Jens Beißwenger, jens.beisswenger@student.uni-tuebingen.de, 2024

Bearbeitungszeitraum: 01.11.2023-30.04.2024

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

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich diese vorliegende Arbeit 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 Arbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Jens Beißwenger (Matrikelnummer 6198682), May 8, 2024

Abstract

Autonomous urban driving exposes numerous corner cases that previous planning approaches struggle to handle robustly. The CARLA Leaderboard 2.0 simulator introduces 38 complex scenarios involving high-speed highway driving, traffic violations, and dynamic obstacles, posing a demanding closed-loop benchmark for autonomous systems. We propose PDM-Lite, the first rule-based expert system to successfully navigate all scenarios in this realistic environment. Integrating components like the Intelligent Driver Model, kinematic bicycle predictions, and dynamic lane changing for obstacle handling, PDM-Lite demonstrates state-of-the-art performance through extensive evaluations. It achieves near-perfect route completion rates, low infractions per kilometer, and driving scores improving upon the previous best method Kyber-E2E by +17.05 points. PDM-Lite establishes a robust and interpretable baseline on the complex scenarios, enabling high-quality data collection for sample-efficient imitation learning. We release PDM-Lite as an open-source system to accelerate research in this critical domain.

Acknowledgments

I would like to thank my supervisors Katrin Renz and Kashyap Chitta for their support during this project. They provided me with ideas, helped me with the CARLA environment, and gave valuable feedback on this report. I also thank Julian, my CARLA Autonomous Driving Challenge teammate, and Bernhard Jaeger, who were part of our weekly meetings for helpful discussions. Finally, I thank Andreas Geiger for the opportunity to work on this project and for his ideas on different controller types and ways to tune this expert for an imitation learning approach.

Contents

1 Introduction

Road traffic accidents are a significant global issue, resulting in the loss of over 1.19 million lives annually, with a majority of these incidents stemming from human errors [28, 43]. Autonomous driving technology holds the potential to substantially reduce these casualties and revolutionize transportation by providing independent mobility for all individuals, including the elderly and disabled. Despite substantial investments and numerous breakthroughs in recent years, the realization of full autonomy in driving remains an ongoing challenge.

Traditionally, the automotive industry has employed modular pipelines to address autonomous driving, encompassing interconnected modules for perception, localization, planning, and control. However, this approach is susceptible to error propagation, as each module heavily relies on the correct outputs from the preceding modules, leading to the propagation of errors throughout the entire system with potentially severe consequences in case of failures. While modular pipelines were initially favored for their interpretability, a trait that end-to-end models lacked, recent advancements in auxiliary outputs [16] and attention maps [30] have mitigated this limitation, paving the way for end-to-end models.

The advent of simulators such as CARLA [11] and HighwayEnv [21] has facilitated the cost-effective generation of vast quantities of ground-truth labels for training end-to-end models. These models are often trained using imitation learning [16] on datasets gathered by expert drivers leveraging privileged information directly from the simulator. While previous rule-based experts could navigate CARLA Leaderboard 1.0 [11] almost perfectly, they struggle with the more challenging CARLA Leaderboard 2.0 [5]. In this updated version, agents face 38 different scenarios (compared to 10 in CARLA Leaderboard 1.0), including higher velocities, longer routes, and obstacles blocking the route requiring lane changes and invading the opposite lane in order to successfully proceed. Additionally, some scenarios require agents to be attentive to actors violating traffic rules, such as running red lights.

In this work, we propose PDM-Lite (Predictive Driver Model-Lite), a rule-based expert algorithm capable of safely navigating an agent in a realistic 3D simulator while handling all 38 scenarios. To the best of our knowledge, this is the first rule-based expert algorithm with this capability developed 18 months after the release of CARLA Leaderboard 2.0 in November 2022.

In Chapter 2, we present related work in the autonomous driving literature, including

modular pipelines and end-to-end imitation learning approaches. We also discuss reinforcement learning approaches and works on expert driver systems. Following this, in Chapter 3, we outline the problem setting and explain the architecture of PDM-Lite and its relevant mechanisms. Subsequently, in Chapter 4, we examine the performance of PDM-Lite on several evaluation settings available in CARLA and provide further ablation studies. Finally, in Chapter 5, we summarize the work and outline potential avenues for future research.

2 Related Work

In this chapter, we examine the existing literature on autonomous driving, focusing on different model architectures and data collection methods. Section 2.1 discusses the two prevalent model architectures, while Section 2.2 explores various data collection techniques.

2.1 Model architectures

The most prominent model architectures in the field of autonomous driving can be broadly classified into two categories: modular pipelines or end-to-end models. Although some hybrid variations exist [20], we will primarily focus on these two.

2.1.1 Modular pipeline

The modular pipeline approach decomposes the problem of autonomous driving into smaller, well-defined subtasks, each with a dedicated input, output representation, and task specification [17]. These modules can be summarized into four main categories. PDM-Lite is a modular approach. However, since it has access to the simulator, it does not require a perception module, and thus, we employ the remaining three stages.

Perception: This module describes the process of fusing raw sensor data (e.g., RGB cameras, LiDARs, Radars, and inertial measurement units) and transforming them into a more meaningful representation, such as bounding boxes. Perception tasks include object detection [23, 26, 47], semantic segmentation [10, 34, 39], lane detection [24, 35], occupancy prediction [37, 42], and object tracking [4, 25, 45, 51].

Prediction: Utilizing the output from the perception modules, this component forecasts the future states of surrounding actors over a fixed time horizon. Due to the complexity of this forecasting task, deep learning approaches are often employed, proposing multiple potential trajectories. In the case of PDM-Lite, we only forecast single trajectories, which proves quite effective, as shown by [31, 44]. Particularly, pedestrians in CARLA exhibit simple behavior, either waiting or walking in a single direction with constant speed.

Planning: This stage determines the ego vehicle's trajectory. Initially, a path planning algorithm (e.g., A*) is used for route planning to find a high-level path to the goal.

Subsequently, behavior and motion planning modules compute the local vehicle trajectory.

Several planning methods propose multiple trajectories and vehicle motions. For instance, Predictive Driver Model (PDM) [9], employs a rule-based predictive planner and a learned ego-forecasting module to generate 15 trajectory proposals. The best proposal is then chosen using a scoring function. In contrast, PDM-Lite uses a single proposal and checks for collisions by forecasting with the kinematic bicycle model.

Control: As the final stage, the control module translates checkpoints into actual vehicle commands, such as steering, throttle, and brake. Common controller types for this stage are proportional-integral-derivative (PID) [19] controllers and model-predictive control (MPC) [13].

Modular pipelines are widely used in industry [27,36,41] due to interpretability and parallel development, but face issues like error propagation from early modules and suboptimal human-chosen output representations, limiting performance. Redundant computation across modules can also lead to inefficiencies.

2.1.2 End-to-end approaches

Unlike modular pipelines, end-to-end approaches use a single neural network to directly map raw sensor data to vehicle control outputs or an equivalent representation. This monolithic architecture learns the entire task end-to-end, avoiding error propagation across modules and leveraging data-driven representations instead of hand-crafted ones.

End-to-end models are favored in research for their potential advantages. They are commonly trained via either imitation learning (IL) from expert demonstrations [16,30,32,33] or reinforcement learning (RL) through environment interactions [22,50].

2.2 Data collection

Data collection is crucial for training autonomous driving systems. Real-world data requires expensive manual labeling by experts across multiple driving scenarios. In research, simulators like CARLA are often used for automated data collection.

2.2.1 Real-world data collection

A notable example of real-world data collection is the nuScenes dataset [3], comprising 15 hours of driving data from 1,000 scenes in Boston and Singapore, manually annotated by human experts according to predefined instructions. Although labour-intensive and costly, this ensures accurate labelled data, which is essential for training models.

2.2.2 Expert drivers in CARLA

For data collection in the CARLA simulator, various expert drivers leveraging privileged simulator information have been developed:

Rule-based experts Rule-based expert systems formulating handcrafted rules from simulator ground truth data have been widely used in CARLA and employed in works like [2, 7, 8, 16, 18, 30, 32, 33, 40, 49].

Transfuser (TF) [8] proposes an ensemble transformer architecture for CARLA Leaderboard 1.0 using imitation learning, which fuses LiDAR and RGB sensor data. It employs a rule-based expert that forecasts other agents using an action-repeat assumption together with the kinematic bicycle model. **PlanT** [30] and **CaT** [48] introduce transformer-based architectures, with PlanT using a simplistic object-level representation and CaT pioneering a knowledge distillation framework for the Teacher-Student paradigm. Both PlanT and CaT employed the expert from TF. Other state-of-the-art IL approaches like **ReasonNet** [33] and **InterFuser** [32] also employ rule-based experts, though without specific details about their expert systems.

DriveCoT [40] introduced an end-to-end driving dataset with chain-of-thought reasoning labels for the CARLA Leaderboard 2.0. The dataset was generated using a rule-based expert system inspired by [15] and TF++ Expert [8], with enhancements for high-speed driving scenarios. The expert system incorporates kinematic bicycle modeling for forecasting, bounding box collision detection, and selects from five discrete target speeds based on leading vehicles. It imposes a 8.33 m/s speed limit in tight turns and employs independent PID controllers for lateral and longitudinal control. In contrast, PDM-Lite forecasts for a shorter 2-second horizon, utilizes the Intelligent Driver Model (IDM) for continuous target speed selection instead of discrete speeds, allows higher speeds of up to 17.78 m/s in turns, and employs a linear regression model for longitudinal control instead of a PID controller.

Kyber-E2E [49], the CARLA Leaderboard 2.0 2023 winner, employs a modular pipeline leveraging language-assisted perception, Inverse Reinforcement Learning for motion planning, and dedicated modules for object and traffic signal detection. It predicts other objects' trajectories using an Unscented Kalman Filter and heuristics, samples up to 132 potential trajectories using Bezier curves and constant acceleration profiles, and selects the best trajectory based on multiple cost functions. In contrast, PDM-Lite forecasts other agents by assuming they repeat their previous action and uses a single path with up to two target speeds: one determined by the Intelligent Driver Model (IDM) and another set to zero when a future collision is detected. PDM-Lite utilizes a PID controller for lateral control and a linear regression module for longitudinal control.

Transfuser++ (TF++) [16] proposes improvements to the expert from TF. It drives faster by introducing higher target speeds of 8 m/s for regular driving, 5 m/s inside

intersections, 2 m/s near pedestrians and stop signs, and 0 m/s for collisions or red lights. Additionally, it utilizes a safety box in front of the ego vehicle, which increases in size, approximating the vehicle's required braking distance. The ego vehicle is forecasted by alternately unrolling longitudinal/lateral PID controllers and the kinematic bicycle model. Collisions are detected if there are any intersections with the ego vehicle, leading to braking.

Building upon the TF++ expert, our work, designed for the CARLA Leaderboard 2.0, introduces several additional innovations. We increase the target speed up to 24 m/s and employ the Intelligent Driver Model (IDM) for continuous target speed selection. Furthermore, we demonstrate that the safety box is not necessary for safe driving, and preemptively decreasing the speed near stop signs and pedestrians like TF++, even before they could be visually perceived, is unnecessary. We modify the forecasting method of the ego vehicle with the kinematic bicycle model. Additionally, we show that a linear regression module for longitudinal control outperforms the PID controller used in TF++.

Reinforcement Learning-based expert systems While rule-based experts are constrained by their designers' choices, reinforcement learning approaches could potentially exceed these limitations, which is often the motivation for employing them.

Think Twice [18] and **TCP** [46] are both imitation learning approaches that adopt Roach [50] as the expert driver, which is a reinforcement learning-based network. Roach utilizes Bird's-Eye View (BEV) images and maps them to continuous low-level actions. To mitigate collisions, they incorporate an additional rule-based vehicle and pedestrian detection module, similar to TF [8].

Think2Drive [22] employs model-based reinforcement learning to train a privileged expert based on the DreamerV3 architecture [14]. They utilized TCP [46] as the sensor-based agent, which was trained on a dataset gathered by the expert.

3 Method

This chapter presents the core components and algorithms that comprise PDM-Lite, designed to solve all 38 scenarios of the CARLA Leaderboard 2.0. We begin by outlining the problem setting and associated challenges in Section 3.1. The necessary preliminaries, including the Intelligent Driver Model (IDM) and kinematic bicycle model, are introduced in Section 3.2. The following sections detail the key stages of the PDM-Lite. Section 3.3.1 describes the strategies for handling route obstacles, while Section 3.3.2 explains the target proposal determination mechanism. Actor forecasting and collision detection are covered in Sections 3.3.3, 3.3.4, and the control command generation is presented in Section 3.3.5.

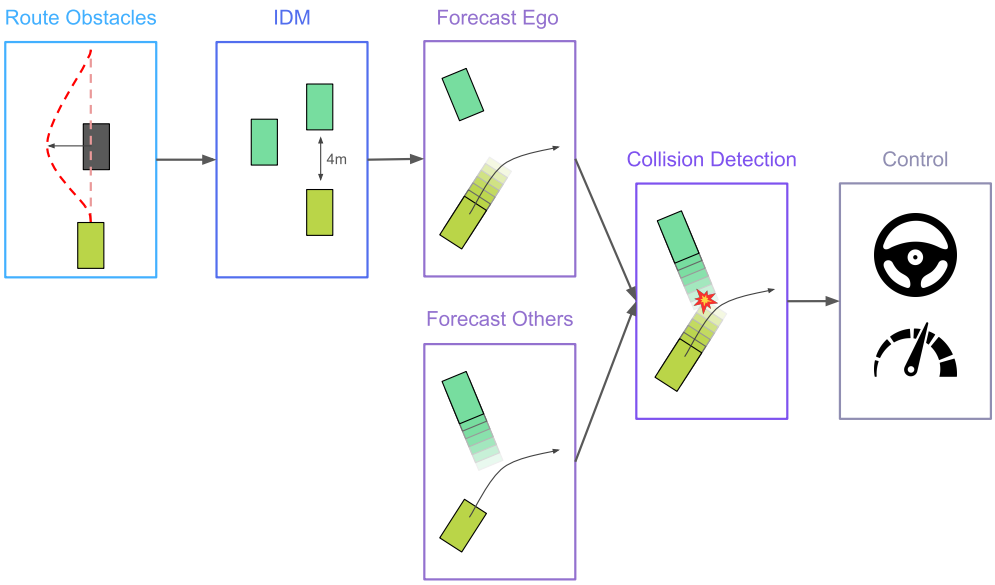

Figure 3.1: PDM-Lite. Our new planner that solves all 38 scenarios of CARLA Leaderboard 2.0, consists of 6 stages.

3.1 Problem setting

The task involves autonomous driving in the realistic 3D CARLA simulator version 0.9.14 [12]. The agent must navigate a predefined route, represented as a sequence

of checkpoints, and reach the destination within a specified time limit. The routes encompass diverse environments, including rural and urban areas, with multiple instances drawn from 38 available scenarios. These scenarios range in complexity, from simple green light intersections to challenging non-signalized turns and high-speed highway maneuvers. The primary objective is to complete the routes safely while adhering to traffic regulations and minimizing violations.

Route Obstacles: Some of these scenario types introduce obstacles that block the agent's path, necessitating dynamic adaptation of the provided route. These so-called RouteObstacles include:

- *Accident*
- *ConstructionObstacle*
- *HazardAtSideLane*
- *ParkedObstacle*
- *AccidentTwoWays*
- *ConstructionObstacleTwoWays*
- *HazardAtSideLaneTwoWays*
- *ParkedObstacleTwoWays*

The "TwoWays" versions require the agent to invade the opposite lane at the right moment, while the normal versions only require a lane change. Additionally, the following scenarios employ similar route adaptation mechanisms and are also included in this section:

- *InvadingTurn*
- *VehicleOpensDoorTwoWays*
- *YieldToEmergencyVehicle*

3.2 Preliminaries

To fully grasp the fundamentals of PDM-Lite, we briefly outline the most crucial concepts in the following subsections. For a more comprehensive understanding of the underlying principles, we refer the reader to the work of [15].

3.2.1 Intelligent Driver Model

The Intelligent Driver Model (IDM) [38] is a widely used car-following model employed to estimate the target velocity when there is a "leading actor" present. The IDM describes the position (x_α) and velocity (v_α) of a vehicle α by maintaining a safe

distance from the leading vehicle ($\alpha - 1$). It uses the net distance ($s_\alpha = x_{\alpha-1} - x_\alpha - l_{\alpha-1}$), which is the distance to the leading vehicle minus the length of the leading vehicle ($l_{\alpha-1}$), and considers the speed difference ($\Delta v_\alpha = v_\alpha - v_{\alpha-1}$) between the two vehicles. The IDM is governed by the following set of ordinary differential equations:

$$\begin{aligned} \dot{x}_\alpha &= \frac{\mathrm{d}x_\alpha}{\mathrm{d}t} = v_\alpha \\ \dot{v}_\alpha &= \frac{\mathrm{d}v_\alpha}{\mathrm{d}t} = a\left(1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right) \\ &\text{with } s^*(v_\alpha, \Delta v_\alpha) = s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}} \end{aligned} \tag{3.1}$$

The IDM parameters, their values, and descriptions are provided in Table 3.1. The values for the minimum desired net distance (s_0) and minimum desired time headway (T) to the leading vehicle depend on the type of leading actor, as discussed in Section 3.3.2.

Parameter	Value	Description
v_0	0.72 v_{lane}	Desired velocity in free traffic
s_0	-	Minimum desired net distance
T	-	Minimum desired time headway to the leading vehicle
a	11 ms^{-2}	Maximum vehicle acceleration
b	20 ms^{-2}	Comfortable braking deceleration
δ	4.0	Acceleration exponent

Table 3.1: IDM parameters, their values, and descriptions used by PDM-Lite. The values for s_0 and T depend on the type of leading actor and are described in Section 3.3.2.

3.2.2 Kinematic Bicycle Model

To accurately predict a vehicle's future motion and prevent collisions, we employ the kinematic bicycle model [29]. This model simplifies the vehicle's four-wheel dynamics by representing it as a two-wheel system moving on a 2D plane, as shown in Figure 3.2. The kinematic bicycle model allows us to compute a vehicle's future position (x_{t+1}, y_{t+1}), velocity v_{t+1}, and orientation ψ_{t+1} at the next time step $t+1$ using Equations 3.2. To estimate the parameters for the front and rear wheelbase lengths (l_f and l_r), as well as the mapping between steering commands and the vehicle's steering angle, we rely on the work by [6]. They collected a dataset of vehicle driving in CARLA and optimized the model's hyperparameters to best fit the observed vehicle dynamics.

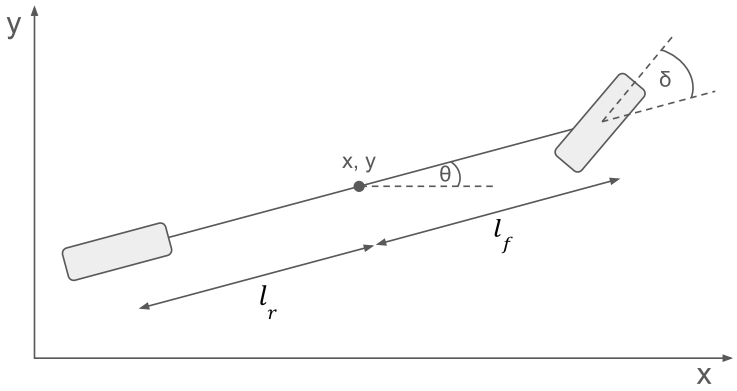

Figure 3.2: **Kinematic Bicycle Model.** The kinematic bicycle model PDM-Lite used to forecast agent behavior in CARLA.

$$
\begin{aligned}
\beta(\delta) &= \arctan\left(\tan(\delta)\frac{l_r}{l_f + l_r}\right) \\
x_{t+1} &= x_t + v_t \cos(\theta_t + \beta(s))\Delta t \\
y_{t+1} &= y_t + v_t \sin(\theta_t + \beta(s))\Delta t \\
v_{t+1} &= v_t + a\Delta t \\
\theta_{t+1} &= \theta_t + \frac{v_t}{l_r}\sin(\beta(\delta))\Delta t
\end{aligned}
\tag{3.2}
$$

Furthermore, we use the parameters from [6] to transform the throttle and brake commands of other vehicles into their corresponding accelerations a_t. For simplicity, we assume that the ego vehicle can either accelerate (when throttle $\neq 0$) or decelerate (when brake = 1). The velocity v_{t+1} while accelerating is estimated using a 2nd-order polynomial based on the current velocity v_t and the throttle value. The velocity v_{t+1} while braking is estimated using a 7th-degree polynomial that depends solely on the velocity v_t. These polynomial models were optimized based on a dataset of the ego vehicle's driving in CARLA. Using different equations to transform brake and throttle commands into acceleration, turns out to be crucial for accurate forecasting.

3.3 PDM-Lite

By integrating the components displayed in Figure 3.1, the PDM-Lite planner aims to safely and efficiently navigate the diverse environments and scenarios of the CARLA Leaderboard 2.0.

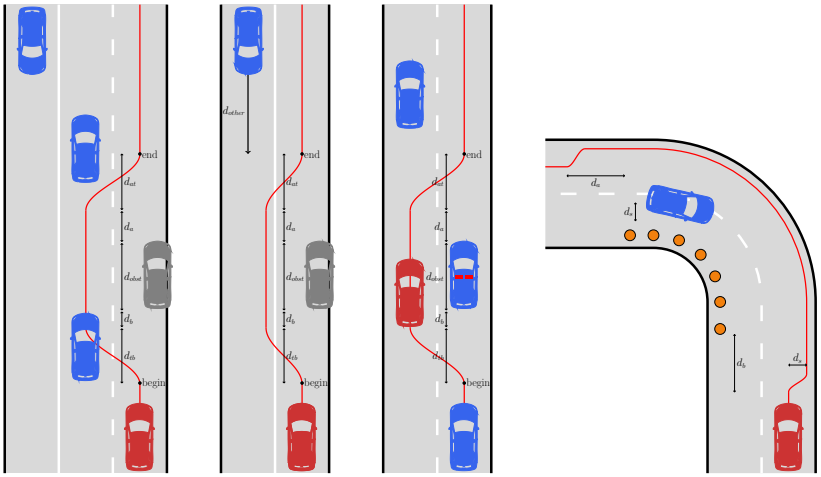

Figure 3.3: The four categories of *RouteObstacles* from left to right: Lane Change Route Obstacles, Opposing Lane Invasion, Conditional Lane Change Maneuvers, Sidewalk Invasion. We display the ego vehicle in red and the other vehicles in blue.

3.3.1 Route obstacle handling

Scenario Tracking

To effectively identify and manage these scenarios, we add them to a list at the moment of their initialization. This list contains all relevant attributes for handling each individual scenario, including the first and the last actor of the obstruction. In scenarios where a single object obstructs the path, such as *ParkedObstacle*, the first and last actors are the same entity.

Route Modification Strategies

The 11 *RouteObstacles* scenarios share a common characteristic: at least one actor blocks the ego vehicle's path, requiring it to shift either to another lane or to the sidewalk. Based on the obstacle locations and the lane change feasibility at these points, the expert determines whether to use the left or right lane. It then calculates the center lane checkpoints and shifts the pre-calculated route by the length d_{obst} of the blocking object to the selected side. To ensure a safe distance from the obstacle, additional distances d_b and d_a are added before and after d_{obst}, respectively. To maintain a smooth path, cosine-based transitions of lengths d_{tb} and d_{ta} are applied before and after the shifted path segment. The specific parameter values are summarized in Table 3.2.

We can distinguish these *RouteObstacles* into four categories:

Lane Change Route Obstacles This category encompasses the scenarios:

- *Accident*
- *ConstructionObstacle*
- *ParkedObstacle*

In all three scenarios, objects on the route obstruct the ego vehicle's path, necessitating a lane change maneuver to a neighboring lane and subsequently returning to the original lane. After shifting the obstructed part of the route, the collision detection (Section 3.3.4) and target speed determination (Section 3.3.2) systems handle the remaining aspects, and these scenarios can be marked as resolved and removed from the list.

Opposing Lane Invasion: The second category contains the scenarios:

- *AccidentTwoWays*
- *ConstructionObstacleTwoWays*
- *HazardAtSideLaneTwoWays*
- *ParkedObstacleTwoWays*
- *VehicleOpensDoorTwoWays*

In contrast to the first category, this category requires the ego vehicle to not only modify its path but also temporarily invade the opposite lane. This involves continuously monitoring the opposing lane's traffic and estimating the appropriate moment for lane invasion, as the time window is often narrow. The expert shifts the part of the route close to the obstacle. For *ParkedObstacleTwoWays*, the path is not fully shifted to the center of the opposing lane, reducing the time spent on that lane.

Until the opposite lane is considered free, PDM-Lite also considers the location marked with 'begin' as a leading actor (Section 3.3.2). To determine whether the opposite lane is free, we compare the time required for vehicles on the opposite lane to traverse the distance d_{other} (assuming they keep their velocity) to the time required for the ego vehicle to reach the 'end' point. If the ego vehicle's time is 200 ms less, the path is considered free, and the target speed determination (Section 3.3.2) and collision detection (Section 3.3.4) computations are overwritten until the 'end' point is reached. Instead, the ego vehicle's target speed is set to 11.11 m/s for *VehicleOpensDoorTwoWays* and 13.88 m/s for all other scenarios.

For *HazardAtSideLaneTwoWays*, which involves two bicycles obstructing the ego vehicle's path, the route is only shifted once the opposing lane is considered free. Until then, the ego vehicle uses its target speed determination system (Section 3.3.2) to follow both bicycles.

Conditional Lane Change Maneuvers The third category is akin to the first category and consists of the following scenarios:

- *HazardAtSideLane*
- *YieldToEmergencyVehicle*

In both of these scenarios, the ego vehicle must execute a lane change maneuver to circumvent the obstruction. In HazardAtSideLane, the obstruction consists of two bicycles, while in YieldToEmergencyVehicle, the ego vehicle must yield to an emergency vehicle approaching from behind. Similar to the first category, PDM-Lite shifts the route to either neighboring lane. For the latter one, a segment of the route 30 meters ahead of the ego vehicle is shifted for a distance of 36 meters to the adjacent lane, to ensure there is enough braking space in case the expert cannot perform the lane change immediately. Unlike the first category, the length of the shifted segment often proves inadequate in both scenarios. To address this, d_a is continually increased until the emergency vehicle passed the ego vehicle, or the ego vehicle passed the bicycles, respectively. Subsequently, the scenario can be removed from the list.

Sidewalk Invasion This category comprises only a single scenario:

- *InvadingTurn*

In this scenario, the ego vehicle must drive on the sidewalk due to an obstruction in the opposite lane, demarcated by construction cones. This forces oncoming traffic to invade the ego vehicle's lane by a distance d_s. To navigate this scenario, PDM-Lite shifts the route close to the construction cones, to the sidewalk, which marks the scenario as completed as the target speed determination system (Section 3.3.2) and the collision detection system (Section 3.3.4) handle the driving, and hence it can be removed from the list.

Scenario	d_{tb}	d_b	d_a	d_{at}
Accident	8	0	0	8
AccidentTwoWays	4	-1.5	-1.5	4
ConstructionObstacle	10	0	0	10
ConstructionObstacleTwoWays	4	1.5	2	4
HazardAtSideLane	8	0	0	8
HazardAtSideLaneTwoWays	8	2	5.5	8
InvadingTurn	0	15	10	0
ParkedObstacle	8	0	0	8
ParkedObstacleTwoWays	4	-0.5	-0.5	4
VehicleOpensDoorTwoWays	4	-2	-2	4
YieldToEmergencyVehicle	8	-	16	8

Table 3.2: Parameter values for the *RouteObstacle* scenarios. Negative values mean d_{obst} is reduced by the corresponding values. For YieldToEmergencyVehicle, d_b does not exist because the route is shifted ahead of the ego vehicle, as the emergency vehicle is approaching from behind.

3.3.2 Target speed proposal with IDM

When a leading actor is present, we employ the Intelligent Driver Model (IDM) to determine the target speed. In its original form, IDM only considers a single leading actor. However, in our approach, we determine the target speed for all potential "leading actors" using IDM and subsequently use the minimum of these target speeds to select the final target speed for the ego vehicle.

Vehicles: To identify if an actor can be considered a leading actor, we calculate its closest location along the expert's planned trajectory. If the distance is less than 2.5 meters and its heading angle is within 35 degrees of the trajectory slope at that location, the actor is considered a leading actor and processed with IDM.

Pedestrians: To detect if a pedestrian is truly crossing the ego vehicle's path, we first forecast their motion using the method described in Subsection 3.3.3. This is necessary because pedestrians sometimes behave erratically and do not start walking or walk in the wrong direction. If we detect a potential collision based on the forecasted motion, we also consider that pedestrian as a leading actor.

Stop Signs and Traffic Lights: Since IDM can also be used for static objects, we consider orange and red traffic lights as leading actors. Similarly, we consider unresolved stop signs as leading actors. Once the ego vehicle's velocity is less than 0.1 m/s and it is closer than 3 meters to the stop sign, we can mark it as resolved and ignore it.

Route Obstacles: We also employ IDM for the *RouteObstacle* scenarios, but for a detailed description, we refer to Section 3.3.1.

As the desired velocity for the ego vehicle, we use 72% of the speed limit. Since other vehicles' target speeds are typically set to 70%, this ensures that the ego vehicle is slightly faster, avoiding unnecessary collisions with trailing vehicles. Inside junctions, we take the minimum of 17.77 m/s and 72% of the speed limit to avoid high velocities at sharp turns.

Table 3.3 summarizes the minimum spacing (s_0) and the desired time headway (T) used in the IDM for different types of leading actors.

Leading object	**Minimum spacing s_0**	**Desired time headway T**
Vehicles & Bicycles	4.0	0.25
Stop signs	2.0	0.5
Traffic lights	5.0	0.5
Walkers	4.0	0.5
Road Obstacles	0.1	0.25

Table 3.3: IDM parameters used by PDM-Lite for different types of leading actors.

3.3.3 Actor forecasting

While the Intelligent Driver Model (IDM) is an effective method for preventing collisions with leading actors, it does not account for potential collisions from all directions and performs poorly during lane changes. Therefore, we employ an additional collision detection mechanism that forecasts the trajectories of all actors, including the ego vehicle, and checks for collision threats. Since the simulator runs at 20 frames per second (FPS), we forecast each actor's trajectory at 50-millisecond intervals. The length of the forecast for all actors is determined by the ego vehicle and is further detailed in the part *Forecast Ego Vehicle*.

Forecast Others

Forecast Other Vehicles To forecast the trajectories of other vehicles, we must consider that their target speeds and routes are unknown. However, since we can access their previous control values, we assume they will apply similar actions in the near future. We then sequentially employ the kinematic bicycle model together with their previous control values to obtain their future bounding boxes. It is important to note that these forecasts do not necessarily align with the road network and often deviate significantly from the vehicles' actual future trajectories. To compensate for this, we linearly increase the size of their bounding boxes by up to 200% as the forecast extends further into the future.

Forecast Pedestrians We forecast pedestrian motion with a constant velocity in their heading direction, as they move without turning. To ensure the ego vehicle safely lets them pass, we increase their speeds to a minimum of 0.5 m/s. Otherwise, they might walk into the ego vehicle, which has to wait due to a blocking actor, causing an infraction penalty. For additional safety, we also increase the size of the pedestrian bounding boxes to 3 meters.

Forecast Ego Vehicle

By forecasting the motion of the ego vehicle, we aim to determine whether there is a risk of collision if the ego vehicle attempts to reach its target speed (0.72 v_{lane}). Since we have complete knowledge of the ego vehicle's trajectory, we can model its motion with greater precision compared to the motion of other vehicles on the road.

To compute the ego vehicle's motion, we alternate between executing the controllers (as described in Section 3.3.5) and the kinematic bicycle model. The kinematic bicycle model simulates the ego vehicle's motion given the estimated control values, while the controllers estimate the control values the expert will likely use in the simulated location based on the route.

The length of the ego vehicle's forecast determines the length of the other actors' forecasts and depends on the ego vehicle's location. If the ego vehicle is close to a

lane change, we forecast for 1.1 seconds. Otherwise, we forecast for 2 seconds. We define "close" as a lane change occurring within $\max\{20, v^2 * 12.96/200 + 10\}$ meters (the maximum of 20 meters and the braking distance plus 10 meters for safety) in the future or if a lane change has already occurred within the past 20 meters.

To account for uncertainty and maintain a safety margin, we increase the forecasted bounding box extents of the ego vehicle by up to 130%.

3.3.4 Collision Detection

After the forecasting phase, we conduct collision checks at each time step to determine if the ego vehicle's forecasted bounding box intersects with that of another actor at the same time step. When the ego vehicle is not in close proximity to a lane change, we filter out vehicles ahead or behind the ego. This is based on the assumption that collisions with leading vehicles are mitigated through speed regulation with the IDM, and that rear vehicles will brake to avoid collisions.

If we detect an intersection of forecasted bounding boxes at the same time step, we set the target speed to 0. For the case where the intersecting bounding box corresponds to a pedestrian, we utilize the IDM to determine the target speed, as explained in Subsection 3.3.2. This is necessary because pedestrians sometimes fail to start walking as expected.

3.3.5 Control commands

Controlling the vehicle requires three primary commands: throttle, brake, and steering. The expert treats lateral control and longitudinal control independently, assuming that the vehicle is not significantly affected by slip, and that the route, including the modified portions for bypassing route obstacles, involves smooth transitions.

Longitudinal control

The longitudinal control of the vehicle is achieved using a simple linear regression model to estimate the throttle and brake values, as described by Equation 3.3. To ensure the throttle remains within its valid range, we clip the computed value between 0 and 1. The input features to the model are polynomial combinations of the vehicle's current velocity (v) and the clipped distance to the target velocity ($\max\{0, v_{target} - v\}$). The specific model parameters ($\mathbf{w}^\top$) were determined using Bayesian optimization on a test route encompassing different target speeds.

$$\begin{aligned} \text{throttle} &= \text{clip}(\mathbf{w}^\top \mathbf{x}, 0, 1) \\ \text{brake} &= 0 \end{aligned} \tag{3.3}$$

Additionally, based on the vehicle's dynamic characteristics, we use the control values described by Equation 3.4 once the error between the current speed (v) and the target speed (v_{target}) exceeds certain thresholds.

$$\begin{aligned} &\text{if } v_{target} - v > 0.53\ m/s: \\ &\qquad \text{throttle, brake} = 1, 0 \\ &\text{if } v/v_{target} > 1.03: \\ &\qquad \text{throttle, brake} = 0, 1 \end{aligned} \tag{3.4}$$

Note that we only apply the full brake value when necessary, rather than using smooth braking values.

This type of model cannot accumulate errors between the current velocity and the target velocity, which might be problematic on slopes. However, we opted against using a Proportional-Integral-Derivative (PID) controller, which includes error accumulation, as it sometimes exhibited abrupt behavior due to the accumulation of errors. An ablation study using a PID controller can be found in Section 4.4.

Lateral control

For lateral control, we employ a Proportional-Integral-Derivative (PID) controller to minimize the difference between the vehicle's heading angle and the angle to a target checkpoint. Since the vehicle's ability to take tight turns decreases as its velocity increases due to slip, we dynamically scale the distance to the target checkpoint based on the vehicle's velocity (v). The scaling factor is expressed as:

$$\text{target distance} = \text{clip}(av + b, 2.4\text{m}, 10.5\text{m}) \tag{3.5}$$

Here, a and b are tunable parameters that were optimized using Bayesian optimization on a test route encompassing a range of speeds and turn types. By dynamically adjusting the target checkpoint distance, we ensure that the vehicle follows a smooth trajectory while accounting for its dynamic limitations at higher velocities. The specific gain parameters for the PID controller were also tuned using Bayesian optimization, and are presented in Table 3.4.

gain parameter	**value**
k_p	3.12
k_d	1.38
k_i	0.64

Table 3.4: Gain parameters of the lateral PID controller.

Special cases

After determining all control values, we also ensure that the vehicle does not roll backward on hills at small distances to the leading vehicle by activating the brake if the throttle value is equal to 0 and the vehicle's velocity is below 0.5 m/s.

In some instances, the vehicle may become blocked by another actor, causing it to stop and wait until the situation is resolved. However, this behavior can lead to failing the ActorBlockTest, which despawns any actor moving slower than 0.1 m/s for at least 180 seconds, resulting in the termination of the entire route. Since the timeout for the ScenarioTimeoutTest (240 seconds) is longer than the ActorBlockTest timeout, we aim to trigger the latter, as it only despawns the vehicles involved in the scenario and often frees the path, resolving the issue. To this end, we count the number of time steps during which the ego vehicle does not move faster than 0.1 m/s, and if this timer exceeds 160 seconds, we accelerate over 0.1 m/s once.

4 Experiments

This chapter describes the experiments conducted to evaluate the performance of PDM-Lite on a variety of different routes. It details the task, routes, metrics, and results obtained by comparing PDM-Lite against various baselines and state-of-the-art methods. The main goal of these experiments is to thoroughly assess PDM-Lite's driving capabilities across diverse scenarios from the CARLA Leaderboard 2.0 and benchmark its performance against existing expert systems and learning-based approaches from recent literature.

4.1 Routes

To evaluate the agent's performance, three collections of routes are provided by the CARLA Leaderboard 2.0: DevTest, Training, and Validation. As some of these routes are up to 13.5 km long and contain 125 scenarios, we split them into smaller routes, each containing only a single scenario. This approach allows a more accurate performance evaluation per scenario type.

4.1.1 Long routes

The term "long routes" refers to the original, unmodified routes. These routes exhibit significant imbalances, with DevTest averaging 7.31 km, Training averaging 8.57 km, and Validation averaging 12.39 km in length. Additionally, the number of scenarios per route varies considerably, with an average of 60, 51, and 93 scenarios in DevTest, Training, and Validation, respectively. It is noteworthy that the Validation routes contain duplicates, with routes 0 to 9 being identical to routes 10 to 19, except for weather conditions. Since PDM-Lite does not use sensor data, its performance is unaffected by weather, and each evaluation of the long Validation routes covers every route twice.

4.1.2 Short routes

To generate a dataset with short routes containing only a single scenario per route, we utilize the original routes mentioned in the previous subsection. We ensure that each route encompasses the full scenario and provides sufficient distance for the vehicle to accelerate as needed by adding appropriate spacing before and after the

scenario. After splitting, the median length of the short routes of the Training routes is 151 m. Subsequently, we randomly sample up to 15 routes per scenario without replacement to create a 15x38 dataset, where 38 represents the number of scenarios. In some cases, fewer routes are available, leading to rare instances with only 1 scenario in case of MergerIntoSlowTrafficV2. In total the 15x38 dataset consists of 500 routes.

4.2 Metrics

To evaluate the expert's performance, we use the three official metrics.

Route Completion (RC) measures the percentage of the route distance completed by the agent. This value is decreased if the agent drives off the road. If the route deviation exceeds 30 meters, the entire route evaluation is terminated. Additionally, the route is stopped if either the agent does not take any action for 180 seconds or if the simulation of a route takes too long to finish, consequently decreasing RC.

Infraction Score (IS) aggregates all types of infractions into a single value. It starts with an ideal score of 1.0, which is multiplied by the corresponding infraction value each time an infraction occurs. This calculation decreases the score exponentially with respect to the number of infractions and can be expressed as a geometric series [1]:

$$IS = \prod_{j}^{\text{ped.,...,stop}} (p_i^j)^{\#\text{infractions}_j} \tag{4.1}$$

Driving Score (DS) measures the overall driving performance of an agent in a single value. It is the product of the route completion and the infraction penalty, RC_iIS_i, and is bounded between 0 and 100, with 100 representing a perfect score. The global driving score, which is the mean of the driving scores of the individual routes (Equation 4.2), serves as the primary metric and classifies participants.

$$DS_{global} = \frac{1}{N} \sum_{i}^{N} RC_iIS_i \tag{4.2}$$

4.3 Results

Baselines To evaluate PDM-Lite's performance, we compare it against **TF++ Expert** [16], **Kyber-E2E** [49], **DriveCoT** [40] and **Think2Drive** [22].

TF++ is an ensemble of transformer-based architectures designed for the CARLA Leaderboard 1.0 using imitation learning. It employs a rule-based expert (TF++

Expert) that forecasts other agents using the action-repeat assumption and the kinematic bicycle model. The ego vehicle is forecasted by alternating between longitudinal/lateral PID controllers and the kinematic bicycle model. It utilizes a safety box in front of the ego vehicle, which expands to approximate the required braking distance. Collisions are detected by intersections with the ego vehicle, triggering braking. The target speeds are 8 m/s for regular driving, 5 m/s at intersections, 2 m/s near pedestrians and stop signs, and 0 m/s for collisions or red lights.

Kyber-E2E [49], the winner of the CARLA Autonomous Driving Challenge in 2023, employed a modular pipeline with language-assisted perception, Inverse Reinforcement Learning for motion planning, and dedicated modules for object and traffic signal detection. It predicted other objects' trajectories using an Unscented Kalman Filter and heuristics, sampled up to 132 potential trajectories using Bezier curves and constant acceleration profiles, and selected the best trajectory based on multiple cost functions. Lateral and longitudinal control utilized independent PID controllers.

DriveCoT [40] presents an end-to-end driving dataset with chain-of-thought reasoning labels, created using the CARLA Leaderboard 2.0 environment. The dataset is generated by an expert rule-based system based on [15], with modifications enabling high-speed driving, dynamic braking distance calculations, and vehicle forecasting via the kinematic bicycle model. The system forecasts 2-3 seconds ahead, checks for collisions using bounding box overlaps, and considers traffic signals and nearby vehicles to select one of five target speeds. A speed limit of 8.33 m/s is imposed in tight turns for safe navigation. Independent PID controllers are used for lateral and longitudinal control. Notably, the dataset routes used by *DriveCoT* are approximately six times shorter than the official Validation routes.

Think2Drive [22] employs model-based reinforcement learning to train a privileged expert based on the DreamerV3 architecture [14] for the CARLA Leaderboard 2.0. As input, it uses multiple bird's-eye view (BEV) semantic segmentation masks, where each channel represents the occurrence of certain types of objects. The network directly outputs the control values for throttle, brake, and steering.

Changes to RunningStopTest We modified the RunningStopTest in the *scenario_runner* module to address an issue with small stop sign trigger boxes. The implementation checks locations up to 4 meters from the stop sign, spaced 0.5 m apart, marking the test as solved if any location is inside the trigger box (vertical purple line in Figure 4.1) and the vehicle is slow enough. However, some trigger boxes are only 2 cm long, causing the test to fail despite stopping correctly, since these locations are unlikely to be inside the trigger box. Our modification increases the trigger box to extent at least 1 meter in each direction (blue rectangle in Figure 4.1), ensuring the test is properly cleared when the vehicle stops as intended.

Table 4.1 presents the performance of PDM-Lite across the different CARLA Leader-

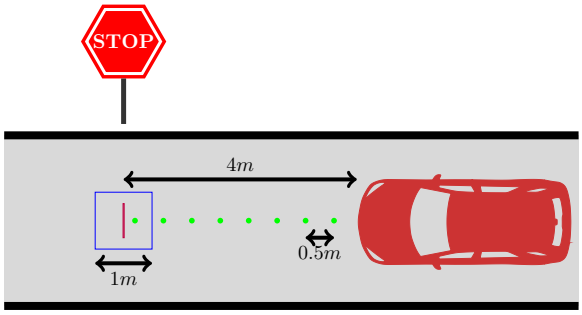

Figure 4.1: Illustration of the modified RunningStopTest implementation, showing the increased trigger box extent (blue rectangle) compared to the original small trigger boxes (vertical purple line).

board 2.0 datasets. To ensure robust evaluation and account for varying route quantities, we use 10 random seeds for the DevTest set, 3 seeds for the Validation set, and 1 seed for the Training set. Since PDM-Lite is a rule-based approach without training, its performance on the Training routes remains unaffected.

Method	Dataset	#Seeds	RC ↑	IS ↑	DS ↑
TF++ Expert	Validation \{3,13}	3	36.3 ±1.9	0.25 ±0.03	2.2 ±0.3
Kyber-E2E	Validation	-	87.1	0.36	27.25
DriveCoT	Validation split to 2km	-	77.5	0.67	51.9
Think2Drive	DevTest	-	98.6	(0.58)	56.8
PDM-Lite (Ours)	DevTest	10	100.0 ±0.0	0.41 / 0.59 ±0.17	40.8 / 58.5 ±16.9
	Validation \{3,13}	3	95.5 ±0.5	0.51 ±0.01	49.3 ±0.8
	Validation	3	85.9 ±0.4	0.46 ±0.01	44.3 ±0.73
	Training	1	98.8	0.49	48.5

Table 4.1: Performance on the long routes showing route completion (RC), infraction score (IS), and driving score (DS) with std for TF++ Expert, Kyber-E2E, DriveCoT, Think2Drive (IS approximated), and PDM-Lite. DriveCoT's performance is reported on 2km splits of the Validation routes. Our results are presented both with and without routes 3 and 13, in which the simulator consistently crashed during evaluation.

Long Routes In all three metrics (RC, IS, and DS), PDM-Lite outperforms TF++ Expert by a significant margin. This is partly due to RouteObstacles, which block the ego vehicle's path and were not present in CARLA Leaderboard 1.0, which TF++ Expert cannot handle. This results in 115 scenario timeouts for TF++ Expert. Furthermore, the traffic speed has increased by almost a factor of 3 to 24 m/s. Scenarios on, i.e. highways require the full speed range, which the baseline cannot

attain, leading to a low DS.

Since routes 3 and 13 consistently crashed during the evaluation setup, they were excluded for a fair comparison with TF++ Expert (Validation \{3,13}). However, to compare with the baseline Kyber-E2E, which included these routes, the worst possible scores (RC=0, IS=0, DS=0) were assumed for PDM-Lite on routes 3 and 13. Even in this scenario, PDM-Lite outperformed the privileged part of Kyber-E2E, the winner of the CARLA Autonomous Driving Challenge in 2023, by a substantial +17.1 driving score (DS) margin. This superior performance stems from a much higher infraction score (IS), indicating fewer infractions, despite Kyber-E2E's slightly higher route completion (RC). Notably, PDM-Lite's RC was lowered by -9.6 points due to the assumed zero RC for routes 3 and 13.

We also include the values achieved by DriveCoT, which obtained a slightly higher DS than PDM-Lite. However, DriveCoT split the official Validation routes into 2km segments, approximately six times shorter than the official routes, so their performance is expected to be significantly lower on the complete official Validation routes and are not directly comparible to PDM-Lites performance.

Comparing PDM-Lite to Think2Drive proves more challenging since the authors have not released their code and full evaluation details. Furthermore, route 1 of DevTest contains an incorrect hit box near a tree close to the street, which causes 2 additional layout collisions for every seed of evaluation. Since DevTest consists of only 2 routes, this limits the final driving score to 71. Hence, we report both IS and DS, with and without these 2 additional detected collisions. In terms of route completion (RC), PDM-Lite achieves consistent route coverage on all 10 seeds and outperforms Think2Drive by a margin of 1.4 points. Although their paper does not report IS and the number of evaluated seeds, we can only approximate their final infraction score with $IS \approx DS/RC$. In case they only evaluated on a single seed, the formula is exact. Assuming they also excluded the collisions with the incorrect hit box, PDM-Lite outperforms Think2Drive by 1.7 points in DS and hence resembles state-of-the-art driving performance to our knowledge. Although using fixed seeds, this route collection is prone to a large driving variance, as indicated by the standard deviation of 16.9.

Furthermore, we also report results on the Validation and Training routes. Since routes 3 and 13 always crashed during evaluation, we report both results with and without them and set their metrics (RC, IS, DS) to 0, which represents the lower bound. The difference in DS between all three datasets can be explained by the differing lengths and the different number of scenarios in each dataset. Also, DevTest only contains 25 out of 38 possible scenario types.

Short Routes To gain a more comprehensive understanding, we report the performance of PDM-Lite in comparison to the TF++ Expert on the 15x38 dataset in Table 4.2. PDM-Lite achieves a perfect RC score of 100.0, completing all routes. The

Scenario	#Routes	TF++ Expert			PDM-Lite			
		RC ↑	IS′ ↑	DS′ ↑	RC ↑	IS′ ↑	DS′ ↑	DS′ Impr. ↑
Accident	15	100.0	0.72	72.0	100.0	1.00	**100.0** ±0.0	+28.0
AccidentTwoWays	15	100.0	0.70	70.0	100.0	1.00	**100.0** ±0.0	+30.0
BlockedIntersection	15	99.0	0.95	93.9	100.0	0.95	**95.5** ±1.2	+1.6
ConstructionObstacle	15	99.8	0.45	45.4	100.0	1.00	**100.0** ±0.0	+54.6
ConstructionObstacle-TwoWays	15	99.9	0.44	44.0	100.0	1.00	**100.0** ±0.0	+56.0
ControlLoss	15	99.5	0.99	98.3	100.0	0.99	**99.1** ±1.5	+0.8
CrossingBicycleFlow	14	100.0	0.93	93.2	100.0	1.00	**100.0** ±0.0	+6.8
DynamicObjectCrossing	15	100.0	1.00	**100.0**	100.0	1.00	**100.0** ±0.0	+0.0
EnterActorFlow	9	100.0	0.82	82.1	100.0	0.93	**92.6** ±2.6	+10.5
EnterActorFlowV2	3	100.0	0.78	77.8	100.0	0.82	**82.2** ±7.7	+4.4
HardBreakRoute	15	100.0	0.99	99.1	100.0	1.00	**100.0** ±0.0	+0.9
HazardAtSideLane	13	100.0	0.68	68.4	100.0	0.99	**99.0** ±1.8	+30.6
HazardAtSideLane-TwoWays	15	99.8	0.71	70.5	100.0	0.96	**96.0** ±1.2	+25.5
HighwayCutIn	10	100.0	1.00	**100.0**	100.0	1.00	**100.0** ±0.0	+0.0
HighwayExit	4	100.0	1.00	**100.0**	100.0	0.97	96.7 ±5.8	-3.3
InterurbanActorFlow	6	100.0	0.93	92.8	100.0	0.98	**98.1** ±3.4	+5.3
InterurbanAdvanced-ActorFlow	6	100.0	0.62	61.9	100.0	0.94	**93.9** ±5.9	+32.0
InvadingTurn	15	100.0	0.72	72.0	100.0	1.00	**100.0** ±0.0	+28.0
MergerIntoSlowTraffic	15	100.0	0.83	83.1	100.0	0.94	**93.8** ±3.1	+10.7
MergerIntoSlowTrafficV2	1	100.0	0.73	73.3	100.0	1.00	**100.0** ±0.0	+26.7
NonSignalizedJunction-LeftTurn	15	100.0	0.85	85.3	100.0	0.99	**98.7** ±2.2	+13.4
NonSignalizedJunction-RightTurn	15	100.0	0.84	84.4	100.0	0.99	**99.1** ±1.5	+14.7
OppositeVehicle-RunningRedLight	15	100.0	1.00	**100.0**	100.0	1.00	**100.0** ±0.0	+0.0
OppositeVehicle-TakingPriority	15	100.0	0.98	98.2	100.0	1.00	**100.0** ±0.0	+1.8
ParkedObstacle	15	99.9	0.71	71.3	100.0	1.00	**100.0** ±0.0	+28.7
ParkedObstacleTwoWays	15	100.0	0.70	70.0	100.0	1.00	**100.0** ±0.0	+30.0
ParkingCrossingPedestrian	15	100.0	0.99	98.7	100.0	1.00	**100.0** ±0.0	+1.3
ParkingCutIn	15	99.9	0.98	97.7	100.0	1.00	**100.0** ±0.0	+2.3
ParkingExit	14	98.5	0.99	97.5	100.0	1.00	**100.0** ±0.0	+2.5
PedestrianCrossing	15	98.9	1.00	98.9	100.0	1.00	**100.0** ±0.0	+1.1
PriorityAtJunction	15	99.7	1.00	99.7	100.0	1.00	**100.0** ±0.0	+0.3
SignalizedJunction-LeftTurn	15	100.0	0.82	82.3	100.0	1.00	**100.0** ±0.0	+17.7
SignalizedJunction-RightTurn	15	100.0	0.86	86.4	100.0	0.98	**98.2** ±1.5	+11.8
StaticCutIn	15	100.0	1.00	**100.0**	100.0	1.00	**100.0** ±0.0	+0.0
VehicleOpensDoor-TwoWays	15	99.9	0.44	44.4	100.0	0.97	**97.3** ±0.0	+52.9
VehicleTurningRoute	15	98.9	0.99	97.8	100.0	1.00	**100.0** ±0.0	+2.2
VehicleTurning-RoutePedestrian	15	99.5	0.97	96.8	100.0	1.00	**100.0** ±0.0	+3.2
YieldToEmergencyVehicle	15	100.0	0.70	70.0	100.0	0.94	**93.5** ±2.7	+23.5
Average	**500**	**99.5**	**0.84**	**83.3**	**100.0**	**0.98**	**98.3** ±0.3	**+10.0**

Table 4.2: Evaluation results of TF++ Expert and PDM-Lite on the 15x38 dataset with 3 seeds. We exclude MinSpeedInfractions and report the infraction score (IS′) and driving score (DS′) without them. Additionally, we report the number of routes and DS′ improvement compared to TF++ Expert.

TF++ Expert also attains an almost perfect RC score due to despawning vehicles after timeouts (163 in total). When the expert cannot manage a scenario after 4 minutes, all scenario-related objects, including vehicles, are despawned, freeing the path. Consequently, the IS for both models is similar to the DS, except for scaling, as $DS = RC \cdot IS$.

We discover that MinSpeedInfractions occur disproportionately often on short routes. Compared to the long Training routes, and considering the 15x38 dataset has 9.3 fewer scenarios, MinSpeedInfractions occur 197 times more frequently, decreasing DS to 92.6. MinSpeedInfractions are calculated for every quarter of the total route and fail if the agent is slower than the average velocity of the background traffic. Since the median length of these routes is 151 meters, MinSpeedInfractions are calculated after driving only 38 meters on average. Hence, any situation where the agent has to wait, such as junctions, stop signs, traffic lights, and almost every scenario, can lead to MinSpeedInfractions.

Consequently, we remove MinSpeedInfractions from IS and DS and report both metrics without them, denoted as IS' and DS'. Table 4.3 provides a detailed list of infraction types. In all scenarios except HighwayExit, PDM-Lite outperforms the TF++ Expert by a substantial margin and often achieves a perfect DS'. For HighwayExit, we assume the result is due to randomness.

The results demonstrate that PDM-Lite struggles the most with scenarios like EnterActorFlow, EnterActorFlowV2, InterurbanAdvancedActorFlow, MergerIntoSlowTraffic, and YieldToEmergencyVehicle. Some of these challenging situations are illustrated in Fig. 4.2. PDM-Lite assumes vehicles repeat their previous action (section 3.3.4), which can cause the forecast to deviate from the actual vehicle motion. Additionally, the binary nature of the collision detection system in PDM-Lite sometimes evaluates the path as free in one timestep, only to detect a collision by intersecting bounding boxes in the next timestep, leading to unnecessary braking or braking too late. In some cases, this leads to actors blocking each other. In other scenarios with higher velocities, this can result in collisions, as seen in the first row of Fig. 4.2. Furthermore, the YieldToEmergencyVehicle scenario requires the ego vehicle to perform a rapid lane change to let the emergency vehicle pass, which is not always successful.

The comparison of infraction types between TF++ Expert and PDM-Lite on the long Validation routes, presented in Table 4.3, further highlights the superior performance of PDM-Lite, especially when considering the much lower RC of TF++ Expert. While PDM-Lite has a higher RC, which increases the chance of infractions, the table shows that it still outperforms the TF++ Expert in most infraction categories. The data indicates that collisions with vehicles account for the majority of infractions. In rare cases, these collisions can lead to additional infractions such as collisions with the layout, outside route lane infractions, or scenario timeouts.

However, as shown in Fig. 4.3, PDM-Lite can also work as intended in the failed routes. With changed traffic manager seeds or slightly different timing, the collisions

Infraction type	Number of infractions ↓	
	TF++ Expert	PDM-Lite
Collisions with layout	1.6	**0.6**
Collisions with pedestrians	**0.0**	0.1
Collisions with vehicles	2.3	**0.9**
Running a red light	0.1	**0.0**
Running a stop sign	0.1	**0.0**
Outside route lanes	**0.0**	0.2
Min speed infractions	0.7	**0.3**
Yield to emergency vehicle infractions	**0.1**	0.2
Scenario timeouts	6.4	**0.5**
Route deviations	**0.0**	**0.0**
Vehicle blocked	0.7	**0.1**
Route timeout	**0.1**	**0.1**

Table 4.3: Infraction comparison between TF++ Expert and PDM-Lite on the long Validation routes.

observed in the problematic instances do not occur.

4.4 Ablation Study

To further evaluate the efficacy of our proposed PDM-Lite algorithm, we conducted an ablation study using the long Validation routes. The results of this study are presented in Table 4.4.

Ablation	RC ↑	IS ↑	DS ↑
Longitudinal PID	94.03	0.46	44.55 ±1.59
IDM $b = 25$	94.82	0.49	46.73 ±5.46
IDM $b = 15$	95.94	0.46	44.33 ±5.75
IDM $a = 13.75$	96.01	0.44	42.49 ±3.81
IDM $a = 8.25$	96.3	0.41	39.23 ±2.36
PDM-Lite	**95.5**	**0.51**	**49.3** ±0.8

Table 4.4: Ablation study results comparing the performance of PDM-Lite with different parameter settings and a PID controller for longitudinal control, using the long Validation routes.

As shown in Table 4.4, the linear regression model for longitudinal control clearly outperforms a PID controller, achieving a 4.75 higher Driving Score (DS). To ensure a fair comparison, we tuned the PID controller's parameters using Bayesian opti-

mization, similar to the linear regression model in PDM-Lite, resulting in only the proportional gain k_p being non-zero.

We also evaluated the impact of varying the maximum acceleration and comfortable braking values by ±25%. For simplicity, we keep these parameters constant in PDM-Lite, although they depend on the current vehicle's velocity. Hence, tuning them is crucial. While PDM-Lite outperforms these ablations, the DS scores of the ablations are still relatively high, indicating that the algorithm is not overly sensitive to these parameters.

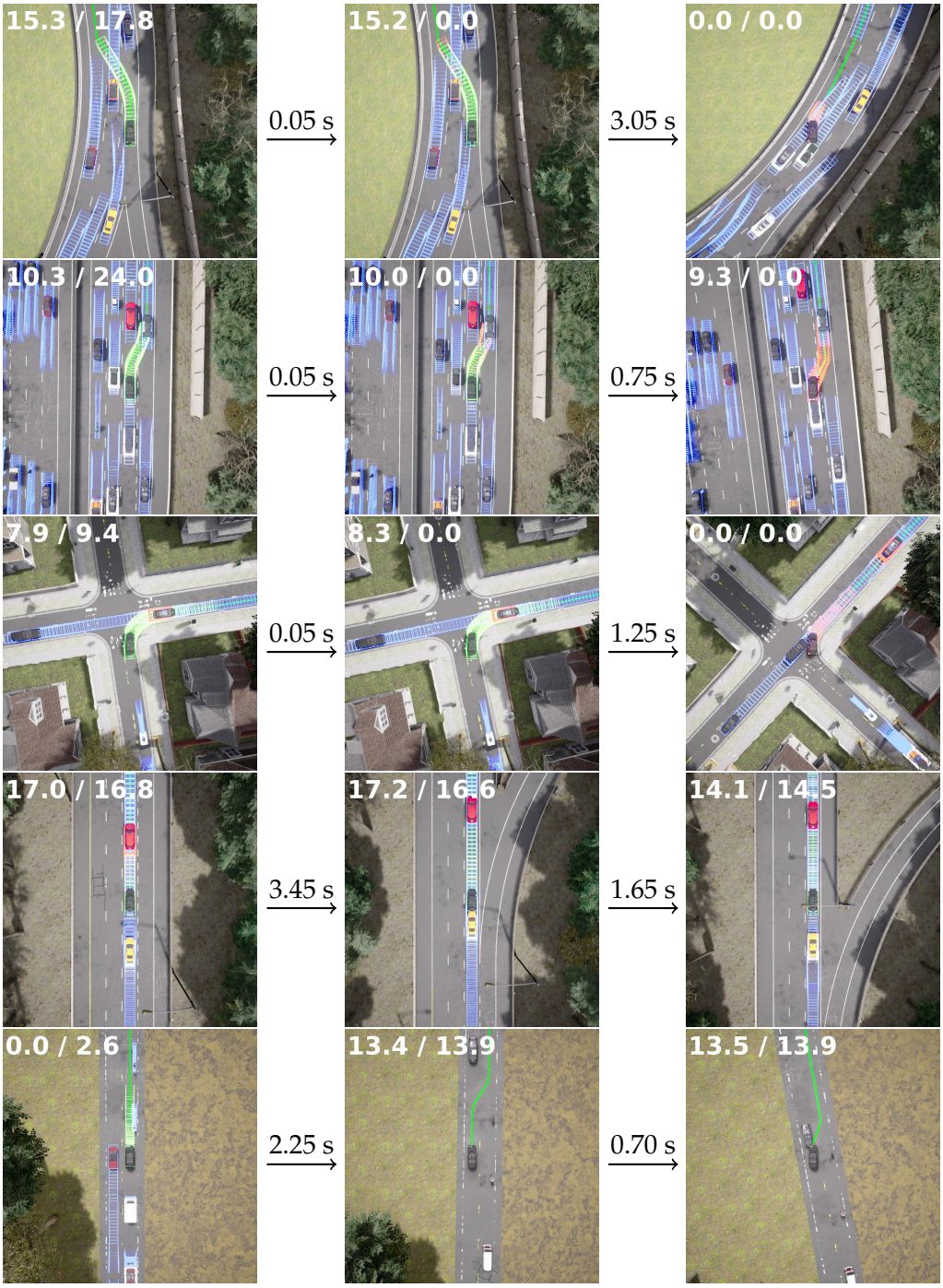

Figure 4.2: PDM-Lite failures in EnterActorFlow, HighwayExit, NonSignalizedJunctionRightTurn, EnterActorFlowV2 and HazardAtSideLaneTwoWays scenarios. Current velocity / target velocity (m/s) are shown in the top left corner. Forecasts deviate from actual predictions, causing braking and blocking or collisions. In blue we display the other vehicles' forecasted bounding boxes and in green the ego vehicle's forecasted bounding boxes. In case we detect intersecting bounding boxes, the subsequent ones are red instead of green.

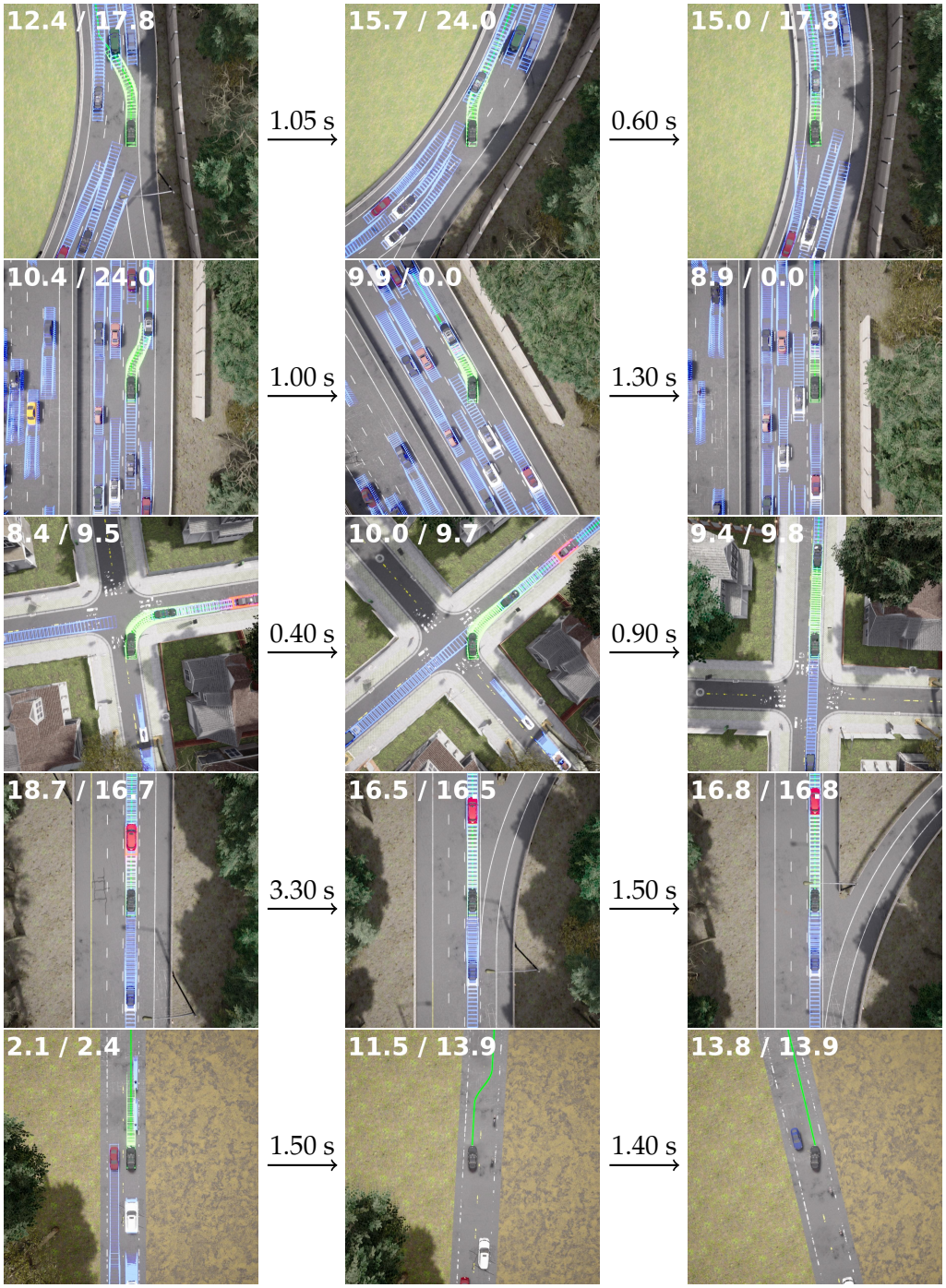

Figure 4.3: Examples of PDM-Lite successfully navigating the EnterActorFlow, HighwayExit, NonSignalizedJunctionRightTurn, EnterActorFlowV2 and HazardAtSideLaneTwoWays scenarios, which it struggled with in Fig. 4.2. Current velocity / target velocity (m/s) are shown in the top left corner. In blue we display the other vehicles' forecasted bounding boxes and in green the ego vehicle's forecasted bounding boxes. In case we detect overlapping bounding boxes, the subsequent ones are red instead of green.

5 Conclusion

The PDM-Lite algorithm presented in this report represents a significant milestone in the field of autonomous driving, demonstrating exceptional performance on the challenging CARLA Leaderboard 2.0 benchmark. Through its modular and interpretable rule-based approach, PDM-Lite surpasses the capabilities of previous expert systems and even the latest learning-based methods in this high-fidelity 3D simulation environment.

The key strength of PDM-Lite lies in its systematic and comprehensive strategy for addressing the diverse challenges encountered in autonomous driving scenarios. By effectively handling route obstacles, proposing appropriate target speeds using the Intelligent Driver Model (IDM), accurately forecasting the motion of other actors, and employing robust collision detection and control mechanisms, PDM-Lite is able to navigate complex urban and highway environments seamlessly while adhering to traffic regulations and minimizing infractions.

The results presented in this report substantiate PDM-Lite's superior performance. On the long Validation routes, PDM-Lite achieves a driving score of 49.3, significantly outperforming the TF++ Expert baseline from CARLA Leaderboard 1.0, which scores only 2.2. Remarkably, PDM-Lite even surpasses the performance of the privileged planner of Kyber-E2E, the winner of the CARLA Autonomous Driving Challenge in 2023, which obtained a driving score of 27.25. Furthermore, on the shorter 15x38 dataset, PDM-Lite attains near-perfect scores, averaging a driving score of 98.3 compared to 83.3 for the TF++ Expert.

While PDM-Lite excels in most scenarios, the results also revealed potential areas for improvement. Scenarios involving merging and entering traffic flow, and yielding to emergency vehicles pose challenges due to limitations in vehicle motion forecasting. Addressing these limitations by incorporating more sophisticated forecasting models or leveraging learning-based planners could further enhance PDM-Lite's performance in such complex situations.

The ablation study highlights the superiority of the linear regression model for longitudinal control over a traditional PID controller. Furthermore, the results showcase PDM-Lite's robustness by exhibiting minimal sensitivity to variations in parameters such as maximum acceleration and comfortable braking.

In conclusion, the PDM-Lite algorithm presented in this report represents a significant stride forward in the pursuit of autonomous driving solutions. Its modular design,

interpretable rule-based approach, and effective integration of various components yield great performance on the demanding CARLA Leaderboard 2.0 benchmark. PDM-Lite paves the way for future advancements in autonomous driving technology and provides a good expert for generating data for end-to-end imitation learning models.

Bibliography

[1] CARLA leaderboard. `https://leaderboard.carla.org/`. Accessed: April 2, 2024.

[2] Tanmay Agarwal, Hitesh Arora, and Jeff Schneider. Affordance-based reinforcement learning for urban driving. *arXiv preprint arXiv:2101.05970*, 2021.

[3] 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 preprint arXiv:1903.11027*, 2019.

[4] Jiarui Cai, Mingze Xu, Wei Li, Yuanjun Xiong, Wei Xia, Zhuowen Tu, and Stefano Soatto. Memot: Multi-object tracking with memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8090–8100, 2022.

[5] CARLA. Carla autonomous driving leaderboard, 2022. [2].

[6] Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. Learning to drive from a world on rails, 2021.

[7] Dian Chen and Philipp Krähenbühl. Learning from all vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17222–17231, 2022.

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

[9] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Predictive driver model: A technical report.

[10] Zichao Dong, Bowen Pang, Xufeng Huang, Hang Ji, Xin Zhan, and Junbo Chen. Lvic: Multi-modality segmentation by lifting visual info as cue. *arXiv preprint arXiv:2403.05159*, 2024.

[11] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017.

[12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[13] 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.

[14] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[15] Bernhard Jaeger. *Expert drivers for autonomous driving*. PhD thesis, Master's thesis, University of Tübingen, 2021. 5.4. 4, 2021.

[16] Bernhard Jaeger, Kashyap Chitta, and Andreas Geiger. Hidden biases of end-to-end driving models, 2023.

[17] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state of the art, 2021.

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

[19] Carl Knospe. Pid control. *IEEE Control Systems Magazine*, 26(1):30–31, 2006.

[20] Eric Leong. Bridging the gap between modular and end-to-end autonomous driving systems. 2022.

[21] Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

[22] Qifeng Li, Xiaosong Jia, Shaobo Wang, and Junchi Yan. Think2drive: Efficient reinforcement learning by thinking in latent world model for quasi-realistic autonomous driving (in carla-v2). *arXiv preprint arXiv:2402.16720*, 2024.

[23] Zhichao Li, Feng Wang, and Naiyan Wang. Lidar r-cnn: An efficient and universal 3d object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7546–7555, June 2021.

[24] Yueru Luo, Chaoda Zheng, Xu Yan, Tang Kun, Chao Zheng, Shuguang Cui, and Zhen Li. Latr: 3d lane detection from monocular images with transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7941–7952, 2023.

[25] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8844–8854, 2022.

[26] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[27] Motional. Voluntary safety self-assessment (vssa), 2021. Online: accessed 31-March-2024.

[28] U NHTSA. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. *DOT HS*, 812:115, 2015.

[29] 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 *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818, 2017.

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

[31] 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.

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

[33] Hao Shao, Letian Wang, Ruobing Chen, Steven L Waslander, Hongsheng Li, and Yu Liu. Reasonnet: End-to-end driving with temporal and global reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13723–13733, 2023.

[34] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021.

[35] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Keep your eyes on the lane: Real-time attention-guided lane detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 294–302, 2021.

[36] Tesla. Tesla ai day 2022, 2022. Online: accessed 31-March-2024.

[37] Xiaoyu Tian, Tao Jiang, Longfei Yun, Yucheng Mao, Huitong Yang, Yue Wang, Yilun Wang, and Hang Zhao. Occ3d: A large-scale 3d occupancy prediction benchmark for autonomous driving. *Advances in Neural Information Processing Systems*, 36, 2024.

[38] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824, August 2000.

[39] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *arXiv preprint arXiv:2111.13260*, 2021.

[40] Tianqi Wang, Enze Xie, Ruihang Chu, Zhenguo Li, and Ping Luo. Drivecot: Integrating chain-of-thought reasoning with end-to-end driving, 2024.

[41] Waymo. Safety report, 2021. Online: accessed 31-March-2024.

[42] Yi Wei, Linqing Zhao, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21729–21740, 2023.

[43] World Health Organization. *Global status report on road safety 2023*. World Health Organization, Geneva, 2023. Licence: CC BY-NC-SA 3.0 IGO.

[44] Haoran Wu, Tran Phong, Cunjun Yu, Panpan Cai, Sifa Zheng, and David Hsu. What truly matters in trajectory prediction for autonomous driving? *arXiv preprint arXiv:2306.15136*, 2023.

[45] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12352–12361, 2021.

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

[47] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[48] Jimuyang Zhang, Zanming Huang, and Eshed Ohn-Bar. Coaching a teachable student. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7805–7815, 2023.

[49] Weize Zhang. Analysis of a modular autonomous driving architecture: The top submission to carla leaderboard 2.0 challenge.

[50] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In

Proceedings of the IEEE/CVF international conference on computer vision, pages 15222–15232, 2021.

[51] Linyu Zheng, Ming Tang, Yingying Chen, Guibo Zhu, Jinqiao Wang, and Hanqing Lu. Improving multiple object tracking with single object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2453–2462, 2021.