2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

GDP: Enhancing End-to-End Autonomous Driving with Goal-Driven Planner

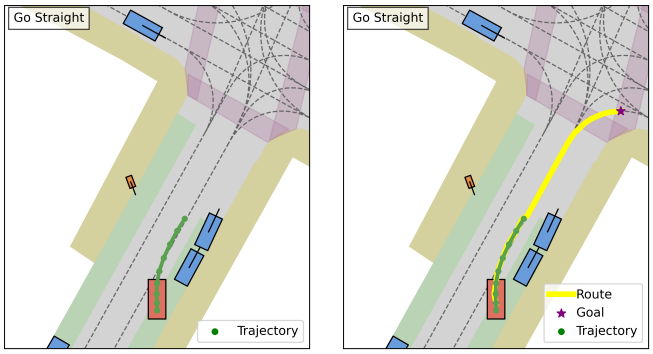
Qiming Zhang, Yue Zhao, Yujian Wang, Wei Wang, Zetong Yang, Wei Xu,
Yin Zhou, Jun Ma, *Senior Member, IEEE*

***Abstract*— End-to-end (E2E) autonomous driving has emerged as a promising paradigm with the pervasive power of model architectures and the availability of large-scale driving datasets. Despite tremendous efforts in recent research, most E2E driving frameworks rely on rather general driving commands, such as "Go Straight" or "Turn Left", which fail to encapsulate the complexities of nuanced driving behaviors and lead to possible semantic ambiguities. Furthermore, such commands are not adequately translated into specific goal locations, which severely limits the planner's capacity to make informed, long-term decisions. This limitation hinders the integration of near-term trajectory planning with long-term goal achievement. To tackle these challenges, we propose the Goal-Driven Planner (GDP), accommodating an appealing plug-and-play feature, which particularly leverages explicit goal points and incorporates two complementary learning objectives: (i) predicting a scene-aware long-term route to the goal, and (ii) refining the near-term trajectory through interaction with the long-term routing. When integrated into off-the-shelf E2E autonomous driving frameworks such as UniAD, VAD-Tiny, and DiffusionDrive, GDP improves trajectory quality and safety across most open-loop and non-reactive simulation metrics. By explicitly modeling a goal-driven route and using it as structured guidance for trajectory refinement, GDP provides a complementary planning signal that enhances long-term goal alignment without modifying the underlying E2E architectures.**

I. Introduction

Recent years have witnessed significant progress in autonomous driving. In particular, end-to-end (E2E) learning is emerging as a promising paradigm, which aims at modeling the direct mapping from sensor inputs to trajectory [1] or the vehicle control. E2E models [2], [3] integrate the classical perception, prediction, and planning into one system, where internal hidden features are shared across tasks. With the unified design, E2E systems can reduce information loss between connected modules, thereby improving system efficiency and performance.

In spite of their significant benefits, existing E2E frameworks for autonomous driving exhibit several limitations concerning navigation. First, most approaches rely on general driving commands like *Go straight*, *Turn left*, and *Turn right*, which are generated by quantizing the relative pose differences of the ego vehicle. Even though these driving commands are relatively easy to implement, they only offer basic semantic instructions and lack geometric precision. As a result, vague semantics may cause ambiguity in driving models, particularly in intricate urban scenarios. As demonstrated in Fig. 1(a), a "Go Straight" instruction could imply a combination of distinct behaviors, such as lane changing or merging. This ambiguity may mislead the planner, potentially resulting in suboptimal planning outcomes or even unsafe driving behaviors. In addition, such high-level instructions fall short in setting a clear goal location for the planner, which is crucial for the model to make long-term decisions. Furthermore, the aforementioned formulation restricts the planning model's ability to learn how to effectively synergize near-term trajectory planning and long-term goal achievement. The lack of an explicit navigation target hinders the model's ability to develop consistent planning strategies connecting local behaviors to overall goal achievement. However, simply embedding static goal points from the map into the E2E planner cannot enhance the performance, as shown in Table I, which still lacks informative instructions for near-term trajectory planning.



(a) Semantic Command (b) Goal-driven Route

Fig. 1: **Comparison of two types of planning guidance for trajectory planning**. (a) **Semantic command** provides high-level driving instructions (e.g., "Go Straight"), with ambiguity. (b) **Goal-driven route** encodes concise objective information and considers surrounding environments, providing more informative instructions.

Method	L2 (m) ↓				Collision (%) ↓			
	1s	2s	3s	Avg.	1s	2s	3s	Avg.
UniAD [3]	0.41	0.63	0.92	0.65	0.05	0.09	0.21	0.12
UniAD‡ [3]	0.43	0.66	0.97	0.69	0.05	0.09	0.26	0.13

Table I: **Driving Command vs. Navigation Points**. ‡ denotes encoding navigation points instead of driving commands in the planner. Metrics follow the VAD [4] settings.

This work was done during the internship at GAC, and supported by the Guangdong provincial project under Grant 2023QN10Z006. *(Project lead: Yin Zhou; Corresponding author: Jun Ma.)*

Qiming Zhang and Jun Ma are with The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511453, China (e-mail: qzhang255@connect.hkust-gz.edu.cn; jun.ma@ust.hk).

Yue Zhao, Yujian Wang, Wei Wang, Zetong Yang, Wei Xu, and Yin Zhou are with GAC R&D Center, Guangzhou 511434, China (e-mail: {zhaoyue, wangyujian, wangwei5, yangzetong5, xuwei, zhouyin}@gac.com.cn).

To address the challenges mentioned above, we propose **GDP** - Goal Driven Planner, a plug-and-play module, which predicts a feasible route path connecting the current position of the ego vehicle to the long-term navigation goal location, simultaneously considering the surrounding environment. Also, GDP consists of a refinement stage, aiming to refine the planning trajectory through interaction with the predicted route. As shown in Fig. 1(b), the learned route smoothly connects the current ego location to the target location and aligns well with the near-term trajectory.

We evaluate our GDP algorithm through extensive experiments on NAVSIM [5] and nuScenes [6]. For experiments over the NAVSIM dataset, we implement GDP based on the recent DiffusionDrive [7], improving the Predictive Drive Model score (PDMS) by 0.9 points, especially in safety metrics and driving efficiency. For experiments over the nuScenes dataset, we integrate GDP into popular frameworks, *i.e.,* UniAD and VAD-Tiny. The open-loop planning results show that our method substantially reduces the average L2 error and the collision rate by 27.4% and 25.0% for UniAD and by 23.1% and 41.5% for VAD-Tiny.

Our contributions can be summarized as follows:

- We thoroughly analyze the ambiguity issue of general driving commands employed by most existing E2E frameworks, and propose navigation goal points with goal-driven routes as a more informative guidance for the planning task in autonomous driving.
- We propose a goal-driven planner - GDP, which comprises a long-term route prediction module and a refinement module to synergize near-term planning and long-term goal achievement. This integration enhances the reliability of routing and also driving safety. It is pertinent to highlight that our GDP accommodates a plug-and-play feature, rendering it compatible with most E2E frameworks.
- Comprehensive evaluations show that GDP enhances the planning performance of E2E models across both open-loop and non-reactive simulation-based metrics, showcasing its strong generalization capability and considerable value in real-world deployment.

II. Related Works

End-to-End Autonomous Driving With the advancement of model architectures, end-to-end autonomous driving has evolved into more integrated and efficient paradigms. ST-P3 [2] adopts a sequential framework that learns to plan trajectories from sensor inputs. UniAD [3] enables joint learning of multiple perception tasks to enhance planning, though it incurs high computational cost. To improve efficiency, VAD [4] introduces a vectorized scene representation, removing the need for rasterization and tracking. Alternative architectural designs have also been explored. PARA-Drive [8] proposes a parallelized structure where perception, prediction, and planning run concurrently. PPAD [9] models agent interactions autoregressively to capture temporal dependencies. DriveTransformer [10] unifies task processing with a transformer decoder, reducing training loss and improving stability. DiffusionDrive [7] incorporates a truncated diffusion process to model multi-modal behaviors efficiently, leading to better planning performance.

Navigation Information in End-to-End Pipeline Navigation information is significant for planning tasks, providing essential instructions for driving decisions at each time step during trajectory planning. Previous works [2], [3] on nuScenes evaluation extract driving commands from the last points of the ground-truth trajectories, and encode them into one-hot vectors to represent driving behaviors, then take them as input for the planning module. VAD [4] and SparseDrive [11] require commands to select specific behavior modes from multi-modal trajectory planning. Moreover, SSR [12] extracts scene representation according to navigation commands, more like a human attention mechanism, bringing higher efficiency and planning performance.

Existing navigation-conditioned E2E planners can be broadly categorized into command-based methods and route-conditioned methods [13]–[15]. Command-based approaches provide abstract semantic intent but lack explicit geometric constraints, which may lead to ambiguity in complex traffic layouts. In contrast, route-conditioned methods leverage spatially grounded navigation information, such as waypoints or map routes, to provide more precise guidance. GDP follows this direction by predicting a learned, goal-driven route as an explicit intermediate representation. Unlike methods that assume access to predefined map routes or closed-loop navigation signals, GDP predicts the route directly from online perception features and sparse goal points, enabling flexible integration with open-loop E2E pipelines.

III. Problem Formulation

A. End-to-End Autonomous Driving

In end-to-end autonomous driving, the primary objective is to generate a feasible trajectory for the ego vehicle, by optimizing the planning loss $L_{planning}$ given the sensor input. To enhance the quality of trajectory learning, auxiliary perception and prediction tasks are jointly optimized, providing richer contextual supervision and promoting more accurate and reliable planning. In our based end-to-end planner, the joint-optimization loss is commonly formulated as:

$$\mathcal{L}_{E2E} = \mathcal{L}_{map} + \mathcal{L}_{agent} + \mathcal{L}_{planning}, \tag{1}$$

where L_{map} is the loss of online mapping [3], [4] or map semantic tasks [7], [16] and L_{agent} is the loss of the agent trajectory prediction head. The map and surrounding agent information is transmitted between different heads in the form of latent features, serving as the input to the planning head, and our GDP modules.

B. Route Prediction Module

Route Pivots We define route pivots as a sequence of key points distilled from aggregated trajectories, representing an accessible path from the current ego vehicle position to the goal. To construct ground truth for training, we

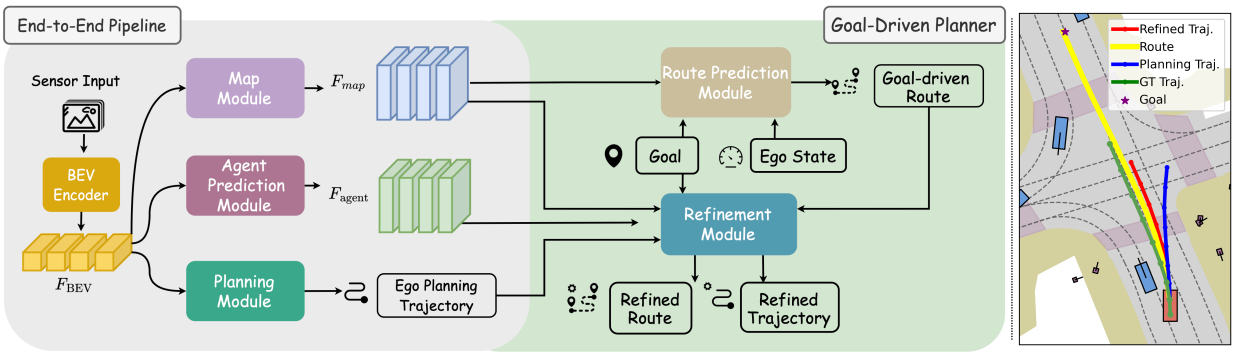


Fig. 2: **Overview of the Goal-Driven Planner (GDP). Left**: GDP extends a standard end-to-end autonomous driving pipeline with a Route Prediction Module and a Refinement Module. The route module predicts a scene-aware route from the ego state, navigation goal, and map features, while the refinement module leverages the predicted route together with map, agent, and initial trajectory information to jointly refine the route and the final planning trajectory. **Right**: An example where the refined trajectory (red) follows the predicted long-term route (yellow) and moves consistently toward the goal (purple star). In contrast, the direct E2E planning output (blue) deviates from the ground-truth trajectory (green) due to the lack of explicit goal and route guidance.

align navigation points with human trajectories and simplify the resulting polyline using the Visvalingam–Whyatt (VW) algorithm [17]. This algorithm reduces redundant points while preserving the overall geometric shape. The process is formulated as:

$$S_{pivot-gt} = F_{VM}(L(P_{trajectory}, P_{goal}), \tau). \quad (2)$$

F_{VM} is the Visvalingam-Whyatt simplifier, and $L(P_{trajectory}, P_{goal})$ means the polyline constructed by the trajectory and the goal point. τ is a threshold, defining a minimum allowable area for the triangles formed by consecutive points in a polyline. Higher τ leads to more sparse extractions, and lower values generate denser route pivots.

Route Prediction Our route prediction module H_{route} takes as input the long-term navigation goal positions P_{goal}, the ego state S_{ego}, and the map features F_{map} extracted from the map module of an E2E pipeline. The objective is to predict a sequence of route pivot points S_{pivot} that guide the downstream planning trajectory refinement module, which can be described as follows:

$$S_{pivot} = H_{route}(P_{goal}, S_{ego}, F_{map}). \quad (3)$$

C. Refinement Module

After learning a goal-driven route from the route prediction module, the refinement module H_{refine} aims to optimize the original trajectory with the predicted route. In the refinement module, it takes the planning trajectory S_{traj} from the original planner, goals P_{goal}, route pivot predictions S_{pivot}, and the scene features (map, agent) as input, and outputs the refined planning trajectory $S_{re-traj}$, as depicted below:

$$S_{re-traj} = H_{refine}(P_{goal}, S_{pivot}, S_{traj}, F_{map}, F_{agent}). \quad (4)$$

IV. Methodology

In this section, we show how to construct navigation points in two representative datasets and the architecture of GDP. As illustrated in Fig. 2, our method complements an E2E pipeline [3], [4], [7] by developing a novel long-term route prediction capability which then provides guidance to refine the ego trajectory. GDP consists of two modules, *i.e.,* the route prediction module and the refinement module, which are explained in Sec. IV-B and Sec. IV-C, respectively. Sec. IV-D shows the overall training loss upon integrating GDP into the E2E pipeline. We discuss the different usage of the learnable route to refine the trajectory.

A. Navigation Point Construction

NAVSIM The NAVSIM dataset does not directly provide map-based navigation points. Instead, driving commands are generated based on static points sampled along the lane centerlines and relative position judgment. To construct static navigation points for our framework, we follow the official methods used in OpenScene [18] for the goal points during generating driving commands[1].

nuScenes Since the nuScenes dataset does not provide navigation points, we construct them manually by extracting semantic locations from the HD map. Inspired by real-world navigation systems (e.g., Google Map), which often highlight intersections, merges, and exits as key areas, we design our navigation points to reflect similar semantics.

Let $\mathcal{S}_{\text{nav}}$ denote the set of navigation points for a trajectory. For each scene and its sampled trajectory, we use the nuScenes map API to retrieve the set of traversed road blocks and extract those corresponding to intersections, denoted by $\mathcal{I}_p$. For each intersection $i \in \mathcal{I}_p$, we compute the intersected points between the polygon boundary of i and the lane centerlines $\mathcal{C}_{\text{lane}}$, represented as linestrings. These intersections form the local navigation points $\mathcal{S}_{\text{nav}}^{\text{local}}$, represented by:

$$\mathcal{S}_{\text{nav}}^{local} = \{\text{Intersect}\,(\mathcal{C}_{\text{lane}}, \text{boundary}(i)) \,|\, i \in \mathcal{I}\}\,. \quad (5)$$

Since the navigation points are typically constructed based on the regions directly traversed by the observed trajectory, they may fail to reflect the vehicle's future progression beyond the

[1] https://github.com/OpenDriveLab/OpenScene/blob/main/DriveEngine/process_data/helpers/driving_command.py

Algorithm 1 Static Navigation Point Construction from nuScenes HD Map

Require: Trajectory sample $\mathcal{T}$, nuScenes HD map $\mathcal{M}$
Ensure: Navigation point set $\mathcal{S}_{\text{nav}}$

```
Initialize S_nav ← ∅
Retrieve traversed road blocks P_rb ← GetRoadBlocks(T, M)
Extract intersections I ← IsIntersection(P_rb)
for each i ∈ I do
    Get lane centerlines C_lane near i
    Compute p_navi ← Intersect(boundary(i), C_lane)
    Add p_navi to S_nav
end for
Locate final point t_end ∈ T
Find nearest lane centerline C_lane^end, trace the forward intersected road intersection i_forward,
Calculate s_nav^fut ← Intersect(boundary(i), C_lane^forward)
Add s_nav^fut to S_nav
return S_nav
```

entire traversal. To address this limitation, we additionally introduce a forward-looking navigation point $\mathbf{s}_{\text{nav}}^{\text{fut}}$, located beyond the trajectory endpoint, to provide guidance that captures the potential future direction and intent. This navigation goal is obtained by finding the closest lane centerline to the final trajectory point and extending it forward to the next intersection boundary. The final navigation set is then:

$$\mathcal{S}_{\text{nav}} = \mathcal{S}_{\text{nav}}^{\text{local}} \cup \{\mathbf{s}_{\text{nav}}^{\text{fut}}\}. \tag{6}$$

B. Route Prediction Module

The route module aims to predict a feasible route as pivot points for refining trajectories, taking the map, agent feature, and the long-term goal as input.

Model Architecture The route module takes a transformer-based learning pipeline as shown in Fig. 3. First, a learnable route embedding F_{route} is randomly initialized. To encode context information, the current ego state and the goal points are fused with F_{route} via vector concatenation. Specifically, the ego state S_{ego}, consisting of the command, velocity, and acceleration information, is processed through MLP to obtain the ego embedding F_{ego}. Similarly, the navigation goal positions $P_{goal} \in \mathbb{R}^{N_g \times 2}$ are flattened and embedded via MLP to produce the goal feature F_{goal}:

$$F_{ego} = \texttt{MLP}(S_{ego}), \quad F_{goal} = \texttt{MLP}(P_{goal}), \tag{7}$$

where N_g denotes the number of goal points. The learnable route embedding, ego embedding, and goal embedding are then concatenated and projected by MLP to form the route query Q_{route}:

$$Q_{route} = \texttt{MLP}(\texttt{Concat}(F_{route}, F_{ego}, F_{goal})). \tag{8}$$

Subsequently, the route query Q_{route} interacts with the map feature F_{map} through a route-map transformer module [19]. In this module, Q_{route} acts as the query, while the map features serve as keys and values, enabling the Q_{route} to learn from spatially relevant information within the scene. An MLP decoder processes the output of the transformer, yielding the final set of route pivot points:

$$S_{pivot} = \texttt{MLP}(\texttt{Transformer}(Q_{route}, F_{map}, F_{map})). \tag{9}$$

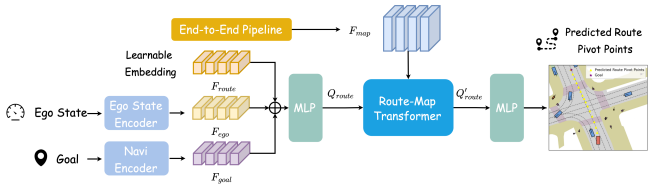


Fig. 3: **Route prediction module**: This module takes as input a learnable route query embedding, concatenating the ego state embedding and goal embedding, then attends the map features from the E2E pipeline to learn route pivot points.

Route Prediction Loss To supervise the predicted route pivots $S_{pivot} \in \mathbb{R}^{N_p \times 2}$, we adopt a dynamic matching strategy inspired by PivotNet [20], aligning them with the ground-truth pivots $S_{gt} \in \mathbb{R}^{N_{gt} \times 2}$ via nearest-point matching. We set $N_p > N_{gt}$. Due to the varying number of ground-truth pivots N_{gt} across samples, the one-to-one supervision is not directly applicable. We therefore match each ground-truth pivot to its nearest predicted pivot to form a matched set $\hat{S}_p^m \in \mathbb{R}^{N_{gt} \times 2}$, and apply an L_1 loss:

$$\mathcal{L}_{match} = \left\| S_{gt} - \hat{S}_p^m \right\|_1. \tag{10}$$

To enforce smoothness and structural consistency, we interpolate the ground-truth pivots, producing an auxiliary target set $S_{p\text{-inter}}^{gt} \in \mathbb{R}^{(N_p - N_{gt}) \times 2}$. Then we apply the same matching process between the remaining $N_p - N_{gt}$ predictions and the auxiliary targets, yielding predictions $\hat{S}_p^{rem} \in \mathbb{R}^{(N_p - N_{gt}) \times 2}$, which is supervised as:

$$\mathcal{L}_{inter} = \left\| S_{p\text{-inter}}^{gt} - \hat{S}_p^{um} \right\|_1. \tag{11}$$

The total route loss is defined as $\mathcal{L}_{route} = \mathcal{L}_{match} + \mathcal{L}_{inter}$, where the first term $\mathcal{L}_{match}$ applies accurate supervision on key pivots, and the second term $\mathcal{L}_{inter}$ strengthens the global structure of the predicted route path.

The predicted route pivots are not intended to represent a globally optimal navigation plan, but rather a scene-aware, goal-aligned geometric guidance that complements the local planning behavior of the base E2E model.

C. Refinement Module

The refinement module does not re-plan from scratch, but performs structured adjustment to the base planner's trajectory by incorporating goal-driven route guidance together with map and agent features. As such, it preserves the local feasibility learned by the base planner while improving long-term alignment.

Model Architecture As depicted in Fig. 4, to further improve the consistency between long-term goal achievement and short-term trajectory planning, GDP jointly refines the predicted route pivots $S_{pivot} \in \mathbb{R}^{N_p \times 2}$ and the original predicted ego trajectory $S_{traj} \in \mathbb{R}^{N_t \times 2}$, where N_t is the

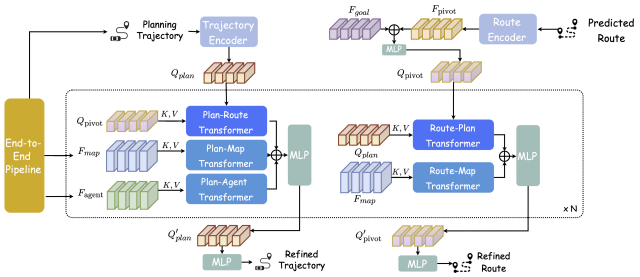


Fig. 4: **Refinement module**: This module consists of two branches for refining route and trajectory, which performs the dual interaction between the learned route and the predicted ego trajectory, considering the surrounding map and agent features.

number of timestamps. Specifically, the refinement module uses an N-stage cascade architecture, with each stage comprising two branches for optimizing route and trajectory, respectively. To refine the trajectory, we encode S_{traj} via self-attention across timestamps and get embedding Q_{plan}. We use Q_{plan} as the query to interact with three different sources of information via dedicated transformers, where the route embedding Q_{pivot}, map feature F_{map} and agent features F_{agent} act as the keys and values in the plan-route, plan-map and plan-agent transformers, respectively. Formally,

$$Q_{plan-*} = \texttt{Transformer}(Q_{plan}, F_*, F_*), \quad * \in \{pivot, map, agent\}, \tag{12}$$

where Q_{plan-*} is the updated planning query after each interaction transformer module. After N-stage refinement, the refined planning trajectory $\hat{S}_{traj} \in \mathbb{R}^{N_t \times 2}$ is obtained as:

$$\hat{S}_{traj} = \texttt{MLP}\big(\texttt{Concat}(Q_{plan-pivot}, Q_{plan-map}, Q_{plan-agent})\big). \tag{13}$$

To refine the route, we use self-attention to encode $S_{pivot} \in \mathbb{R}^{N_p \times 2}$ into F_{pivot} and concatenate it with the goal feature F_{goal}. We project the fused feature into embedding Q_{pivot}. In the route-plan and route-map transformers, we set Q_{pivot} as the query to interact with Q_{plan}, F_{map}, as:

$$Q_{pivot-*} = \texttt{Transformer}(Q_{pivot}, F_*, F_*), \quad * \in \{plan, map\}, \tag{14}$$

where $Q_{pivot-*}$ is the pivot query after each interaction. Finally, we get the refined route pivot points $\hat{S}_{pivot} \in \mathbb{R}^{N_P \times 2}$ as:

$$\hat{S}_{pivot} = \texttt{MLP}(\texttt{Concat}(Q_{pivot-plan}, Q_{pivot-map})). \tag{15}$$

Refinement Loss In the refinement module, we use ground-truth route S_{gt} and the human trajectory $S_{traj-gt}$ as training supervision signals. The losses consist of the route loss $L_{re-route}$ and the planning loss $L_{re-plan}$, same as the planning loss from the original E2E pipeline [3], [4], [7]. The loss is calculated as:

$$\mathcal{L}_{refine} = \lambda_{re-route}\mathcal{L}_{re-route} + \lambda_{re-plan}\mathcal{L}_{re-plan}, \tag{16}$$

where $\lambda_{re-route}$ and $\lambda_{re-plan}$ are loss weights for the route-refinement and planning-refinement branch, respectively.

Hyperparameter	DiffusionDrive	UniAD	VAD-Tiny
Dataset	NAVSIM v1.0	nuScenes	nuScenes
Learning Rate	8e-4	2e-4	2e-4
Batch Size	128	1	4
Epochs	100	20	60
Route Pivot Number N_p	16	16	16
VW Threshold τ	1	10	10
Refinement Layers N	3	3	3
$\lambda_{re-route}$	2.0	1.0	1.0
$\lambda_{re-plan}$	13.0	1.0	1.0
λ_1	1.0	1.0	1.0
λ_2	10	1.0	1.0
λ_3	1.0	1.0	1.0

Table II: **Hyperparameter settings for nuScenes and NAVSIM experiments**. For each E2E method, the auxiliary-task loss weights remain at the same values used in the original implementation.

D. End-to-End Training

By integrating GDP into an E2E framework [3], [4], [7], we train our GDP module with the E2E framework from scratch, maintaining the original training pipeline of the integrated E2E methods. The latent features F_{map}, F_{agent}, F_{BEV} shared in the E2E pipeline are optimized jointly by the auxiliary tasks and GDP modules. The overall training objective inherits the losses from the E2E framework and includes the newly proposed route prediction loss and refinement loss. The overall loss is:

$$\mathcal{L} = \lambda_1\mathcal{L}_{E2E} + \lambda_2\mathcal{L}_{route} + \lambda_3\mathcal{L}_{refine}, \tag{17}$$

where λ_1, λ_2, λ_3 are weights for each task. Also, the E2E loss $\mathcal{L}_{E2E}$ is the combination of the multi-task loss in the original E2E pipeline, computed as: $\mathcal{L}_{E2E} = \sum_{i=1}^{N_{task}} \mathcal{L}_{e2e-task}$. Joint training optimizes shared representations for both long-term and short-term objectives, leading to more coherent and goal-driven planning behavior.

V. Experimental Results

A. Dataset, Baseline Models, and Metrics

NAVSIM dataset evaluates algorithms in the non-reactive simulation setting with closed-loop metrics [5]. It is a re-distribution of nuPlan [21]. We choose **DiffusionDrive** [7] as our baseline and integrate our method. The overall evaluation metric is the predictive drive model score (PDMS), which is an aggregation of multiple sub-scores: no at-fault collisions (NC), drivable area compliance (DAC), time-to-collision (TTC), comfort (Comf.), and ego progress (EP) [7]. nuScenes dataset [6] is widely adopted by the end-to-end autonomous driving studies for open-loop evaluation. To verify the effectiveness of our method, we choose the popular **UniAD** [3] and a more recent **VAD-Tiny** [4] as our baselines. We choose VAD-Tiny as a representative lightweight E2E planner to evaluate the generalization of GDP across different model capacities. The planning performance is assessed by distance error (L2) and box collision rate. We report results using the VAD Metrics [4].

To evaluate the route prediction, we evaluate the geometric alignment between predicted and ground-truth route paths using the projection average distance error (Proj.ADE) and Chamfer distance [22].

Method	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
Transfuser [16]	97.7	92.8	92.8	100	79.2	84.0
UniAD* [3]	97.8	91.9	92.9	100	78.8	83.4
PARA-Drive* [8]	97.9	92.4	93.0	99.8	79.3	84.0
DRAMA [23]	98.0	93.1	94.8	100	80.1	85.5
VADv2 [24]	97.2	89.1	91.6	100	76.0	80.9
Hydra-MDP-C [25]	98.7	98.2	95.0	100	**86.5**	**91.0**
DiffusionDrive [7]	98.2	96.2	94.7	100	82.2	88.1
Goalflow [26]	98.4	**98.3**	94.6	100	85.0	90.3
DiffusionDrive ‡	98.0	95.9	94.3	100	81.7	87.5
GDP (DiffusionDrive)	98.9	96.1	**95.9**	100	83.5	89.0

Table III: **Comparison on NAVSIM-V1.0 dataset-navtest split with PDMS metrics**. $*$ denotes only camera input, and other methods consider camera and lidar features. ‡ means the results of re-implementation with the official settings.

B. Implementation Details

We implement GDP and jointly train the whole E2E pipeline from scratch with 8 NVIDIA RTX A100 GPUs for all experiments. The detail settings are shown in Table II.

C. Quantitative Comparison

Quantitative Comparison on NAVSIM As shown in Table III, integrating the proposed GDP module into DiffusionDrive leads to improvements across most metrics, with particularly notable gains in safety-related aspects such as NC and TTC. The overall planning score increases from 88.1 to 89.0, with notable gains in NC (+0.7), TTC (+1.2), and EP (+1.3), clearly demonstrating the effectiveness of our framework under the standard navigation goal setting. In particular, improvements in safety-related metrics such as NC and TTC indicate that our method produces safer and more collision-averse trajectories, with the assistance of the predicted goal-driven route.

Quantitative Comparison on nuScenes For the nuScenes, simply including the ego-state can lead to obvious improvement [27], [28] in the open-loop metrics. To isolate the effect of route-based guidance and avoid confounding factors introduced by ego-state inputs, we exclude ego-state information for both the baselines and GDP in the nuScenes experiments. As shown in Table IV, GDP also yields consistent improvements relative to stronger baselines, indicating that the observed gains are not solely due to weaker starting points. This setting follows prior studies and ensures a fair comparison under a controlled architecture. Without loss of generality, we drop F_{ego} when constructing the route query Q_{route} in (8). In Table IV, we compare GDP with state-of-the-art end-to-end autonomous driving methods. Integrating GDP into UniAD and VAD-Tiny consistently enhances planning performance over the original baselines. Specifically, GDP reduces UniAD's L2 error by 27.4% and collision rate by 25.0%. For VAD-Tiny, it achieves 23.1% reduction in the L2 error and 41.5% reduction in collision risk. These results demonstrate that our GDP improves the trajectory quality and safety based on different methods, achieving competitive performance with stronger baselines.

D. Ablation Studies

Refinement Module Design The ablation results in Table V demonstrate the effectiveness of each component

Method	L2 (m) ↓				Collision (%) ↓			
	1s	2s	3s	Avg.	1s	2s	3s	Avg.
UniAD [3]	0.45	0.70	1.04	0.73	0.05	0.09	0.21	0.12
VAD-Tiny [4]	0.41	0.75	1.19	0.78	0.26	0.35	0.63	0.41
VAD-Base [4]	0.40	0.72	1.16	0.76	0.07	0.17	0.41	0.21
BEV-Planner [27]	0.27	0.54	0.90	0.57	-	-	-	-
Gen-AD [29]	0.28	0.49	0.78	0.52	0.08	0.14	0.34	0.19
SSR [12]	**0.18**	0.36	**0.63**	**0.39**	**0.01**	**0.04**	**0.12**	**0.06**
DriveTransformer [10]	0.19	**0.34**	0.66	0.40	0.03	0.10	0.22	0.11
GDP (UniAD)	0.36	0.52	0.70	0.53	0.04	0.05	0.17	0.09
GDP (VAD-Tiny)	0.34	0.58	0.89	0.60	0.15	0.25	0.31	0.24

Table IV: **Comparison on nuScenes dataset with open-loop metrics**. No ego-state information is included in the models. We follow the VAD metrics settings.

Refinement Module				NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
Goal	Route	Map	Agent						
-	-	-	-	98.0	95.9	94.3	100	81.7	87.5
✓	-	-	-	98.3	95.4	94.8	100	82.0	87.7
✓	✓	-	-	98.7	95.4	95.9	100	82.7	88.4
✓	✓	✓	-	98.8	95.8	95.6	100	83.0	88.8
✓	✓	✓	✓	**98.9**	**96.1**	**95.9**	**100**	**83.5**	**89.0**

Table V: **Ablation study on the structure of the GDP refinement module**. Goal, route, map, and agent denote whether the corresponding information is incorporated into the refinement module.

in the proposed GDP refinement module. As more modules—the plan-route, plan-map, and plan-agent modules are incorporated, performance consistently improves across all metrics. Specifically, adding the goal-driven route already yields substantial gains in ego progress (+0.7) and safety-related metrics, including NC (+0.4) and TTC (+1.1). Further introducing map and agent interactions leads to additional improvements, with the full model achieving the best results in the PDMS metrics. Introducing sparse navigation points into the refinement planner only brings limited improvement in trajectory quality and can even degrade DAC, as the lack of structured guidance may lead trajectories beyond the drivable area. These trends validate the benefit of combining long-term route, spatial context, and dynamic agent information for structured trajectory refinement.

Route Task Scale Factor	Route Prediction		NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
	Chamf. Dis.↓	Proj. ADE↓						
0.001	2.32	0.40	98.5	95.6	95.4	100	82.5	88.2
0.01	2.01	0.21	98.7	95.5	95.7	100	82.7	88.4
0.1	1.90	0.17	98.8	95.7	95.6	100	83.0	88.7
1	**1.82**	**0.14**	**98.9**	**96.1**	**95.9**	**100**	**83.5**	**89.0**

Table VI: **Correlation between route prediction quality and refined trajectory**. Route prediction is evaluated with Chamfer distance, projected ADE.

Route-Planning Performance Correlation To investigate the relationship between route prediction and trajectory planning, we vary the loss weight of the route prediction task and analyze its impact on trajectory performance. In detail, we adjust the λ_2 in (16) and λ_{route} in (17) by multiplying these parameters with the scale factor from 0.001 to 1, to change the learning quality of route prediction. As shown in Table VI, better route prediction leads to improved planning outcomes, particularly in safety-related metrics such as DAC. This underscores the importance of high-quality route guidance in enabling safe and consistent trajectory

planning.

E. Latency Analysis

To evaluate computational efficiency, we trained the DiffusionDrive, GDP-DiffusionDrive model for 100 epochs on 8 NVIDIA A100 GPUs. The inference performance was subsequently benchmarked on 1 NVIDIA A100 GPU with batch size 1. A detailed breakdown of the associated costs is presented in Table VII. GDP-Diff introduces slightly higher inference latency (33.2 ms) compared to DiffusionDrive (23.9 ms), mainly due to the additional routing (1.3 ms) and refinement (6.5 ms) modules. Nevertheless, the overall latency remains within real-time limits and is accompanied by improved decision-making performance (PDMS 89.0 vs. 87.5), indicating that the performance gains well justify the modest computational overhead.

Method	Training (8×A800)		Inference (1×A100)					
	Batch size	Time (h)	L (ms)	L-R (ms)	L-Re (ms)	Params (M)	FPS	PDMS
DiffDrive	128	2.8	23.9	–	–	60.7	42	87.5
GDP-Diff	128	4.7	33.2	1.3	6.5	78.2	30	89.0

Table VII: **Training and inference cost comparison**. L means the latency. L-R and L-Re represent the computational cost of two involving modules (route, refinement) of GDP.

F. Qualitative Analysis

Fig. 5 illustrates the visualization results demonstrating how GDP improves planning behavior under various scenarios. As shown in Fig. 5, the route prediction can capture basic maneuvers such as a left turn, enabling the planner to generate navigation-aligned trajectories. In Fig. 5(a)-(d), the goal-driven route guidance in GDP enables the vehicle to produce more human-like and goal-consistent behaviors. In contrast, DiffusionDrive, when influenced by ambiguous driving commands, can yield either aggressive or insufficient driving strategies, like scenarios in Fig. 5(c)-(d), where it may lead to potentially unsafe maneuvers. Fig. 5(e) shows that GDP follows the predicted route more efficiently in the safe environment, while DiffusionDrive deviates and slows down. Fig. 5(f) illustrates that GDP can simultaneously avoid potential risks and align with route guidance, whereas DiffusionDrive strictly follows the general command and tends to produce aggressive unsafe maneuvers in crowded scenes. These examples show that GDP not only resolves ambiguity in high-level instructions but also bridges long-term intent with near-term planning.

VI. Discussion

In modern autonomous driving systems, e.g., L4 robotaxi and ADAS-capable vehicle, the navigation route is typically calculated by a standalone software. Although the route provides useful prior information, we argue that the proposed learning paradigm is still highly desired for the following rationales. First, the navigation software relies on accurate real-time localization as well as up-to-date maps with detailed road structure information. Adverse conditions, including poor GPS signal coverage and ad-hoc construction zones, can cause the navigation software vulnerable. Our GDP predicts the route leveraging only the online perception feature and a goal point, which is robust to the aforementioned factors. Second, the navigation software is independent of E2E model training. The proposed paradigm aims to help the E2E framework develop both long-term path finding and near-term planning capabilities via joint optimization. We believe our paradigm better mimics how humans perform strategic tasks, compared to the existing E2E training paradigm.

VII. Limitations and Future Works

We propose a novel learning paradigm, GDP, that generates a semantically enriched navigation route driven by static goal points. These paths provide more informative guidance, leading to improved planning performance and safety. Our method is built upon the assumption that static navigation points are available and aligned with the intended driving direction. These points serve as critical guidance for generating route paths that are both behaviorally meaningful and contextually aware of surrounding traffic. However, in some scenarios where navigation points are misaligned or do not reflect the driver's actual intent, such as free driving or exploratory behavior, our route predictor may generate paths that diverge from human driving patterns. In future work, we plan to extend GDP to more E2E architectures and evaluate its robustness on fully closed-loop datasets, further verifying its scalability and applicability.

References

[1] P. S. Chib and P. Singh, "Recent advancements in end-to-end autonomous driving using deep learning: A survey," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 103–118, 2023.

[2] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. Tao, "St-P3: End-to-end vision-based autonomous driving via spatial-temporal feature learning," in *European Conference on Computer Vision*, 2022, pp. 533–549.

[3] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.

[4] B. Jiang, S. Chen, Q. Xu, B. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang, "VAD: Vectorized scene representation for efficient autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8340–8350.

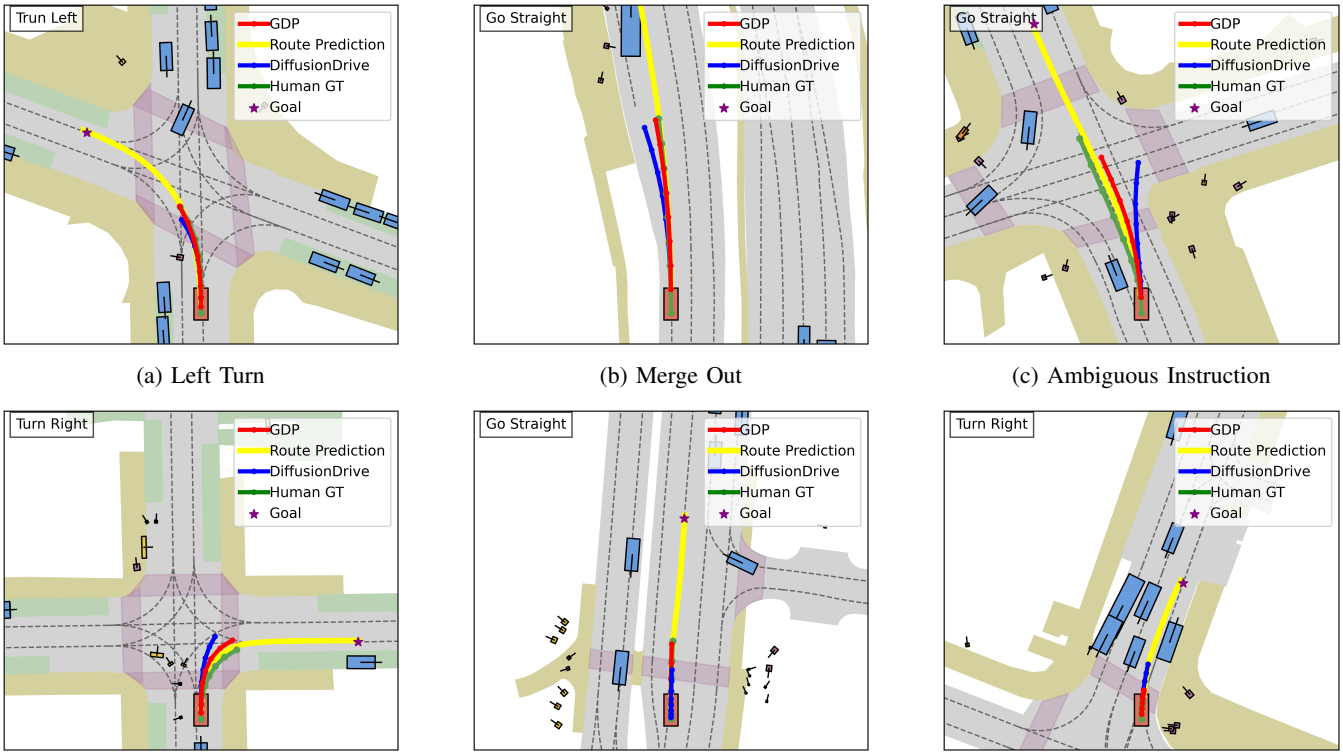
[5] D. Dauner, M. Hallgarten, T. Li, X. Weng, Z. Huang, Z. Yang, H. Li, I. Gilitschenski, B. Ivanovic, M. Pavone *et al.*, "NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking," *Advances in Neural Information Processing Systems*, vol. 37, pp. 28 706–28 719, 2025.

[6] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 621–11 631.

[7] B. Liao, S. Chen, H. Yin, B. Jiang, C. Wang, S. Yan, X. Zhang, X. Li, Y. Zhang, Q. Zhang *et al.*, "Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving," *arXiv preprint arXiv:2411.15139*, 2024.

[8] X. Weng, B. Ivanovic, Y. Wang, Y. Wang, and M. Pavone, "ParaDrive: Parallelized architecture for real-time autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15 449–15 458.

[9] Z. Chen, M. Ye, S. Xu, T. Cao, and Q. Chen, "PPAD: Iterative interactions of prediction and planning for end-to-end autonomous driving," in *European Conference on Computer Vision*, 2024, pp. 239–256.



(a) Left Turn (b) Merge Out (c) Ambiguous Instruction

(d) Insufficient Right-turn (e) Driving Efficiency (f) Avoid Potential Collision

Fig. 5: **Qualitative demonstrations of GDP on NAVSIM**. Route predictions (yellow), GDP-refined trajectory (red), baseline DiffusionDrive prediction (blue), goal points (purple stars), and human trajectories (green) are visualized in several scenarios.

[10] X. Jia, J. You, Z. Zhang, and J. Yan, "DriveTransformer: Unified transformer for scalable end-to-end autonomous driving," *arXiv preprint arXiv:2503.07656*, 2025.

[11] W. Sun, X. Lin, Y. Shi, C. Zhang, H. Wu, and S. Zheng, "SparseDrive: End-to-end autonomous driving via sparse scene representation," *arXiv preprint arXiv:2405.19620*, 2024.

[12] P. Li and D. Cui, "Navigation-guided sparse scene representation for end-to-end autonomous driving," *arXiv preprint arXiv:2409.18341*, 2024.

[13] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta, "Parting with misconceptions about learning-based vehicle motion planning," in *Conference on Robot Learning*, 2023, pp. 1268–1281.

[14] J. Cheng, Y. Chen, and Q. Chen, "PLUTO: Pushing the limit of imitation learning-based planning for autonomous driving," *arXiv preprint arXiv:2404.14327*, 2024.

[15] J. Cheng, Y. Chen, X. Mei, B. Yang, B. Li, and M. Liu, "Rethinking imitation-based planners for autonomous driving," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 14 123–14 130.

[16] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 12 878–12 895, 2022.

[17] M. Visvalingam and J. D. Whyatt, "Line generalization by repeated elimination of points," in *Landmarks in Mapping*, 2017, pp. 144–155.

[18] S. Peng, K. Genova, C. Jiang, A. Tagliasacchi, M. Pollefeys, T. Funkhouser *et al.*, "Openscene: 3d scene understanding with open vocabularies," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 815–824.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[20] W. Ding, L. Qiao, X. Qiu, and C. Zhang, "Pivotnet: Vectorized pivot learning for end-to-end hd map construction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3672–3682.

[21] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "nuPlan: A closed-loop ml-based planning benchmark for autonomous vehicles," *arXiv preprint arXiv:2106.11810*, 2021.

[22] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, and D. Lin, "Balanced chamfer distance as a comprehensive metric for point cloud completion," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 088–29 100, 2021.

[23] C. Yuan, Z. Zhang, J. Sun, S. Sun, Z. Huang, C. D. W. Lee, D. Li, Y. Han, A. Wong, K. P. Tee *et al.*, "DRAMA: An efficient end-to-end motion planner for autonomous driving with mamba," *arXiv preprint arXiv:2408.03601*, 2024.

[24] S. Chen, B. Jiang, H. Gao, B. Liao, Q. Xu, Q. Zhang, C. Huang, W. Liu, and X. Wang, "VADv2: End-to-end vectorized autonomous driving via probabilistic planning," *arXiv preprint arXiv:2402.13243*, 2024.

[25] Z. Li, K. Li, S. Wang, S. Lan, Z. Yu, Y. Ji, Z. Li, Z. Zhu, J. Kautz, Z. Wu *et al.*, "Hydra-Mdp: End-to-end multimodal planning with multi-target hydra-distillation," *arXiv preprint arXiv:2406.06978*, 2024.

[26] Z. Xing, X. Zhang, Y. Hu, B. Jiang, T. He, Q. Zhang, X. Long, and W. Yin, "GoalFlow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving," *arXiv preprint arXiv:2503.05689*, 2025.

[27] Z. Li, Z. Yu, S. Lan, J. Li, J. Kautz, T. Lu, and J. M. Alvarez, "Is ego status all you need for open-loop end-to-end autonomous driving?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 14 864–14 873.

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

[29] W. Zheng, R. Song, X. Guo, C. Zhang, and L. Chen, "GenAD: Generative end-to-end autonomous driving," in *European Conference on Computer Vision*, 2024, pp. 87–104.