

GoalFlow: Goal-Driven Flow Matching for Multimodal Trajectories Generation in End-to-End Autonomous Driving

Zebin Xing^{1,2*}, Xingyu Zhang^{2*}, Yang Hu², Bo Jiang^{4,2}

Tong He⁵, Qian Zhang², Xiaoxiao Long³, Wei Yin^{2†}

¹School of Artificial Intelligence, University of Chinese Academy of Sciences

²Horizon Robotics

³Nanjing University

⁴Huazhong University of Science & Technology

⁵Shanghai AI Laboratory

Abstract

We propose GoalFlow, an end-to-end autonomous driving method for generating high-quality multimodal trajectories. In autonomous driving scenarios, there is rarely a single suitable trajectory. Recent methods have increasingly focused on modeling multimodal trajectory distributions. However, they suffer from trajectory selection complexity and reduced trajectory quality due to high trajectory divergence and inconsistencies between guidance and scene information. To address these issues, we introduce GoalFlow, a novel method that effectively constrains the generative process to produce high-quality, multimodal trajectories. To resolve the trajectory divergence problem inherent in diffusion-based methods, GoalFlow constrains the generated trajectories by introducing a goal point. GoalFlow establishes a novel scoring mechanism that selects the most appropriate goal point from the candidate points based on scene information. Furthermore, GoalFlow employs an efficient generative method, Flow Matching, to generate multimodal trajectories, and incorporates a refined scoring mechanism to select the optimal trajectory from the candidates. Our experimental results, validated on the Navsim[7], demonstrate that GoalFlow achieves state-of-the-art performance, delivering robust multimodal trajectories for autonomous driving. GoalFlow achieved **PDMS of 90.3**, significantly surpassing other methods. Compared with other diffusion-policy-based methods, our approach requires only a **single denoising step** to obtain excellent performance. The code is available at <https://github.com/YvanYin/GoalFlow>.

1. Introduction

Since UniAD[15], autonomous driving has increasingly favored end-to-end systems, where tasks like mapping and

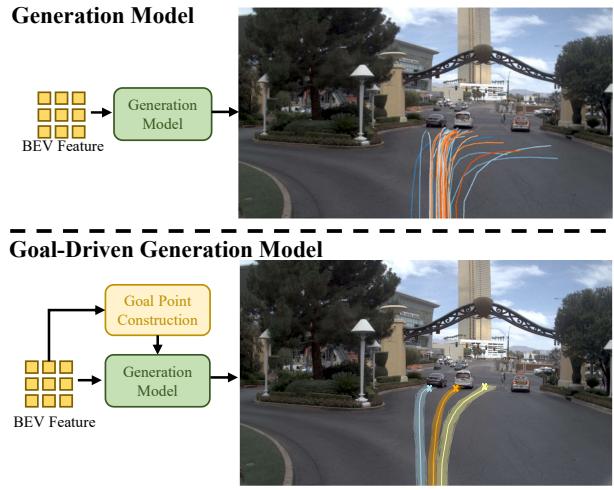


Figure 1. **The comparison of different multimodal trajectory generation paradigms recently.** A standalone generative model often produces highly diverse trajectories with no clear boundaries between different modalities. In contrast, the Goal-Driven Generation Model leverages the strong guidance of goal points, effectively distinguishing multiple modalities by utilizing different goal points.

detection ultimately serve the planning task. To enhance system reliability, some end-to-end algorithms[16, 17, 27] have begun exploring ways to generate multimodal trajectories as trajectory candidates for the algorithms. In autonomous driving, command typically includes indicators for left, right, and straight actions. VAD[17] uses this command information to generate multimodal trajectories. Goal points, which provide the vehicle's location information for the next few seconds, are commonly used as guiding information in other approaches, such as SparseDrive[27]. These methods pre-define a set of goal points to generate different trajectory modes. Both approaches have succeeded in autonomous driving, offering candidate trajectories that significantly reduce collision rates. However, since these methods' guiding information does not pursue accuracy but instead provides a set of candidate values for the trajec-

*Equal contribution.

†Corresponding author, project leader. Email: yvanwy@outlook.com

tory, when the gap between the guiding information and the ground truth is large, it is prone to generating low-quality trajectories.

In recent trajectory prediction works, some methods[18, 28, 32] aim to generate multimodal trajectories through diffusion, using scene or motion information as a condition to produce multimodal trajectories. Other methods [12] utilize diffusion to construct a world model. Without constraints, approaches like Diffusion-ES[32] tend to generate divergent trajectories, which is depicted in the second row of Fig.1, requiring a scoring mechanism based on HD maps to align with the real-world road network, which is difficult to obtain in end-to-end environments. MotionDiffuser[18] addresses trajectory divergence by using the ground truth endpoint as a constraint, which introduces overly strong prior information. GoalGAN[8] first predicts the goal point and then uses it to guide the GAN network to generate trajectories. However, GoalGAN employs grid-cell to sample goal points, which does not consider the distribution of the goal points.

Reviewing previous work, we identified some overlooked issues:(1) Existing end-to-end autonomous driving systems tend to focus heavily on collision and L2 metrics, often adding specific losses or applying post-processing to reduce collision, while overlooking whether the vehicle remains within the drivable area. (2) Most end-to-end methods are based on regression models and aim to achieve multimodality by using different guiding information. However, when the guiding information deviates significantly from the ground truth, it can lead to the generation of low-quality trajectories.

GoalFlow can be divided into three parts: Perception Module, Goal Point Construction Module, and Trajectory Planning Module. In the first module, following transfuser[3], images and LiDAR are fed into two separate backbones and fused into BEV feature finally. In the second module, GoalFlow establishes a dense vocabulary of goal points, and a novel scoring mechanism is used to select the optimal goal point that is closest to the ground truth goal point and within a drivable area. In the third module, GoalFlow uses flow matching to model multimodal trajectories efficiently. It conditions scene information and incorporates stronger guidance from the selected goal point. Finally, GoalFlow employs a scoring mechanism to select the optimal trajectory. Compared to directly generating trajectories with diffusion, as in the first row of Fig. 1, our approach provides strong constraints on the trajectory, leading to more reliable results.

We conducted experimental validation in Navsim and found that our method outperformed other approaches in overall scoring. Notably, due to our goal point selection mechanism, we achieved a significant improvement in DAC scores. Additionally, we observed that this flow-matching-

based approach is robust to the number of denoising steps during inference. Even with only a single denoising step, the score dropped by only 1.6% compared to the optimal case, enhancing the potential for real-world deployment of generative models in autonomous driving.

Our contributions can be summarized as follows:

- We designed a novel approach to establishing goal points, demonstrating its effectiveness in guiding generative models for trajectory generation.
- We introduced flow matching to end-to-end autonomous driving and seamlessly integrated it with goal point guidance.
- We developed an innovative trajectory selection mechanism, using shadow trajectories to further address potential goal point errors.
- Our method achieved state-of-the-art results in Navsim.

2. Related Work

2.1. End-to-End Autonomous Driving

Earlier end-to-end autonomous driving approaches[5][4] used imitation learning methods, directly extracting features from input images to generate trajectories. Later, Transfuser[3] advanced by fusing lidar and image information during perception, using auxiliary tasks such as mapping and detection to provide supervision for the perception. FusionAD[33] took Transfuser a step further by propagating fused perception features directly to the prediction and planning modules. Other methods [19, 20] align the traffic scene with natural language. UniAD[15] introduced a unified query design that made the framework ultimately planning-oriented. Similarly, VAD[17] focused on a planning-oriented approach by simplifying perception tasks and transforming scene representation into a vectorized format, significantly enhancing both planning capability and efficiency. Building on this, some methods[1, 22] discretized the trajectory space and constructed a trajectory vocabulary, transforming the regression task into a classification task. PARA-Drive[30] performs mapping, planning, motion prediction, and occupancy prediction tasks in parallel. GenAD[35] employed VAE and GRU for temporal trajectory reconstruction, while SparseDrive[27] progressed further in the vectorized scene representation, omitting denser BEV representations. Compared to previous methods that focus on better fitting ground truth trajectories using a regression model, we concentrate on generating high-quality multimodal trajectories in an end-to-end setting.

2.2. Diffusion Model and Flow Matching

Early generative models always used VAE[21] and GAN[10] in image generation. Recently, diffusion models that generate images by iteratively adding and remov-

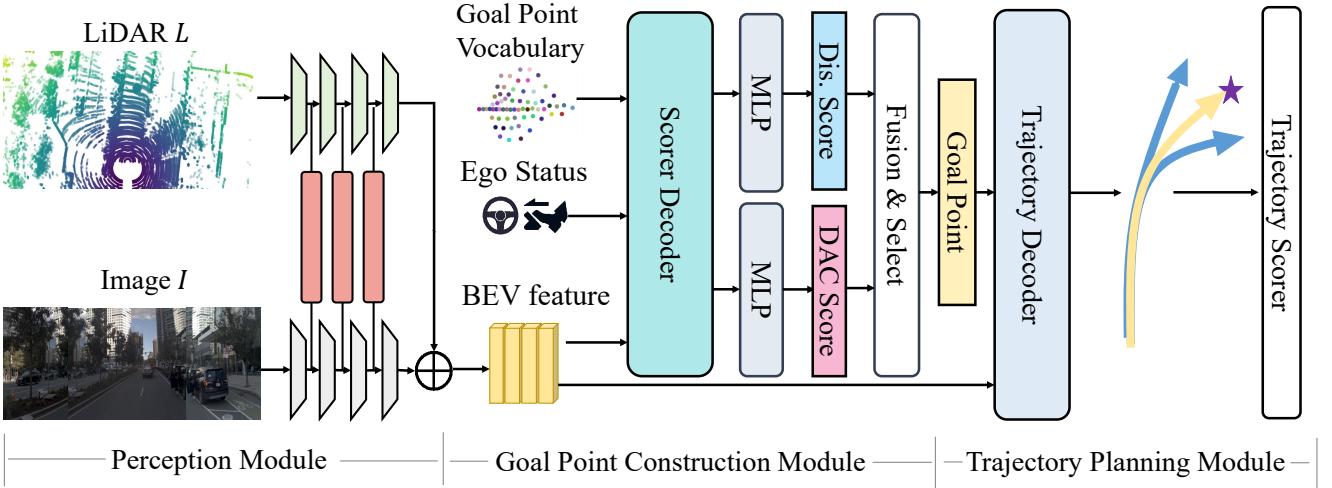


Figure 2. **Overview of the GoalFlow architecture.** GoalFlow consists of three modules. The Perception Module is responsible for integrating scene information into a BEV feature F_{bev} , the Goal Point Construction Module selects the optimal goal point from Goal Point Vocabulary \mathbb{V} as guidance information, and the Trajectory Planning Module generates the trajectories by denoising from the Gaussian distribution to the target distribution. Finally, the Trajectory Scorer selects the optimal trajectory from the candidates.

ing noise have become mainstream. DDPM[14] applies noise to images during training, converting states over time steps, and subsequently denoises them during testing to reconstruct the image. More recent methods[26] have further optimized sampling efficiency. Additionally, CFG[13] has enhanced the robustness of generated outputs. Flow Matching[23] establishes a vector field for transitioning from one distribution to another. Rectified flow[24], a specific form of flow matching, enables a direct, linear transition path between distributions. Compared to diffusion models, rectified flow often requires only a single inference step to achieve good results.

2.3. MultiModal Trajectories Generation

In planning tasks, such as manipulation and autonomous driving, a given scenario often offers multiple action options, requiring effective multimodal modeling. Recent works[2, 31] in manipulation have explored this by applying diffusion models with notable success. Autonomous driving has adopted two main multimodal strategies: the first uses discrete commands to guide trajectory generation, such as in VAD[17], which produces three distinct trajectory modes, and SparseDrive[27] and [16], which cluster fixed navigation points from datasets for trajectory guidance. The second approach introduces diffusion models directly to generate multimodal trajectories[18, 29, 32], achieving success in trajectory prediction but facing challenges in end-to-end applications. Building on diffusion models, we address limitations in accuracy and efficiency by incorporating flow matching, using goal points to guide trajectories with precision rather than focusing solely on multimodal diversity.

3. Method

3.1. Preliminary

Compared to diffusion, which focuses on learning to reverse the gradual addition of noise over time to recover data, flow matching[23] focuses on learning invertible transformations that map between data distributions. Let π_0 denote a simple distribution, typically the standard normal distribution $p(x) = \mathcal{N}(x|0, I)$, and let π_1 denote the target distribution. Under this framework, rectified flow[24] uses a simple and effective method to construct the path through optimal transport[25] displacement, which we choose as our Flow Matching method.

Given x_0 sampled from π_0 , x_1 sampled from π_1 , and $t \in [0, 1]$, the path from x_0 to x_1 is defined as a straight line, meaning the intermediate status x_t is given by $(1 - t)x_0 + tx_1$, with the direction of intermediate status consistently following $x_1 - x_0$. By constructing a neural network v_θ to predict the direction $x_1 - x_0$ based on the current state x_t and time step t , we can obtain a path from the initial distribution π_0 to target distribution π_1 by optimizing the loss between $v_\theta(x_t, t)$ and $x_1 - x_0$. This can be formalized as:

$$v_\theta(x_t, t) \approx \mathbf{E}_{x_0 \sim \pi_0, x_1 \sim \pi_1} [v_t | x_t] \quad (1)$$

$$\mathcal{L}(\theta) = \mathbf{E}_{x_0 \sim \pi_0, x_1 \sim \pi_1} [\|v_\theta(x_t, t) - (x_1 - x_0)\|_2] \quad (2)$$

3.2. GoalFlow

3.2.1. Overview

GoalFlow is a goal-driven end-to-end autonomous driving method that can generate high-quality multimodal trajectories. The overall architecture of GoalFlow is illustrated in

Figure 2. It comprises three main components. In the Perception Module, we obtain a BEV feature F_{bev} that encapsulates environmental information by fusing camera images I , and LiDAR data L . The Goal Point Construction Module focuses on generating precise guidance information for trajectory generation. It accomplishes this by constructing a goal point vocabulary $\mathbb{V} = \{g_i\}^N$, and employing a scoring mechanism to select the most appropriate goal point g . In the Trajectory Planning Module, we produce a set of multi-modal trajectories, $\mathbb{T} = \{\hat{\tau}_i\}^M$, and then identify the optimal trajectory τ , through a trajectory scoring mechanism.

3.2.2. Perception Module

In the first step, we fuse image and LiDAR data to create a BEV feature, F_{bev} , that captures rich road condition information. A single modality often lacks crucial details; for example, LiDAR does not capture traffic light information, while images cannot precisely locate objects. By fusing different sensor modalities, we can achieve a more complete and accurate representation of the road conditions.

We adopt the Transfuser architecture [3] for modality fusion. The forward, left, and right camera views are concatenated into a single image $I \in \mathbb{R}^{3 \times H_1 \times W_1}$, while LiDAR data is formed as a tensor $L \in \mathbb{R}^{K \times 3}$. These inputs are passed through separate backbones, and their features are fused at different layers using multiple transformer blocks. The result is a BEV feature, F_{bev} , which comprehensively represents the scene. To ensure effective interaction between the ego vehicle and surrounding objects, as well as map information, we apply auxiliary supervision to the BEV feature through losses derived from HD maps and bounding boxes.

3.2.3. Goal Point Construction Module.

In this module, we construct a precise goal point to guide the trajectory generation process. Diffusion-based approach[18, 32] without constraints often leads to excessive trajectory divergence, which complicates trajectory selection. Our key observation is that a goal point contains a precise description of the short-term future position, which imposes a strong constraint on the generation model. As a result, we divide the traditional Planning Module into two steps: first, constructing a precise goal point, and second, generating the trajectory through planning.

Goal Point Vocabulary. We aim to construct a goal point set that provides candidates for the optimal goal point. Traditional goal-based methods[11, 34], rely on lane-level information from HD map to generate goal point sets for trajectory prediction. However, HD maps are expensive, making lane information often unavailable in end-to-end driving. Inspired by VADv2[1], we discretize the endpoint space of trajectories to generate candidate goal points, enabling a solution without relying on HD maps. We clustered trajectory endpoints $\mathbf{p}_i = (x_i, y_i, \theta_i)$ in the training data

to create N cluster centers, which form our goal point vocabulary \mathbb{V} . Each endpoint p_i represents a position (x_i, y_i) and heading θ_i . To ensure that the vocabulary represents finer-grained locations, we typically set N to a large value, generally 4096 or 8192.

Goal Point Scorer. High-quality trajectories typically exhibit the following characteristics: A small distance to the ground truth and within the drivable area. To achieve this, we evaluate each goal point g_i in the vocabulary \mathbb{V} using two distinct scores: the *Distance Score* $\hat{\delta}_{\text{dis}}^{\text{dis}}$ and the *Drivable Area Compliance Score* $\hat{\delta}_{\text{dac}}^{\text{dac}}$. The Distance Score measures the proximity between the goal point g_i and the endpoint of ground truth trajectory g^{gt} , with a continuous value in the range $\hat{\delta}_{\text{dis}}^{\text{dis}} \in [0, 1]$, where a higher value indicates a closer match to g^{gt} . The Drivable Area Compliance Score ensures that the goal point lies within the drivable area, using a binary value $\hat{\delta}_{\text{dac}}^{\text{dac}} \in \{0, 1\}$, where 1 indicates that the goal point is valid within the drivable area, and 0 indicates it is not.

To construct the target distance score δ_i^{dis} , we utilize the softmax function to map the Euclidean distance between the goal point g_i and the ground truth goal point g^{gt} to the interval $[0, 1]$. This is defined as:

$$\delta_i^{\text{dis}} = \frac{\exp(-\|g_i - g^{\text{gt}}\|_2)}{\sum_j \exp(-\|g_j - g^{\text{gt}}\|_2)} \quad (3)$$

For the target drivable area compliance score δ_i^{dac} , we introduce a shadow vehicle, whose bounding box is determined based on the position and heading (x_i, y_i, θ_i) in g_i and the shape of the ego vehicle. Let $\{p^j\}^4$ represent the set of four corner positions of the shadow vehicle, and let \mathbb{D} denote the polygon representing the drivable area. The drivable area compliance score δ_i^{dac} is defined as:

$$\delta_i^{\text{dac}} = \begin{cases} 1, & \text{if } \forall j, p^j \in \mathbb{D} \\ 0, & \text{otherwise} \end{cases}$$

We compute the final score $\hat{\delta}_i^{\text{final}}$ by aggregating $\hat{\delta}_i^{\text{dis}}$ and $\hat{\delta}_i^{\text{dac}}$. The goal point with the highest final score is selected for trajectory generation.

$$\hat{\delta}_i^{\text{final}} = w_1 \log \hat{\delta}_i^{\text{dis}} + w_2 \log \hat{\delta}_i^{\text{dac}}$$

As shown in Fig.3(a), the Transformer-based Scorer Decoder uses the result of adding F_v and F_{ego} as the query, with F_{bev} as the key and value. The output is passed through two separate MLPs to produce the scores $\hat{\delta}_{\text{dis}}^{\text{dis}}$ and $\hat{\delta}_{\text{dac}}^{\text{dac}}$ for each point in the \mathbb{V} . Fig.3(b) shows the distribution of these two scores. With the points in warmer colors representing higher scores, we observe that score $\hat{\delta}_{\text{dis}}^{\text{dis}}$ effectively indicates the desired future position, while $\hat{\delta}_{\text{dac}}^{\text{dac}}$ identifies if the goal point is within the drivable area.

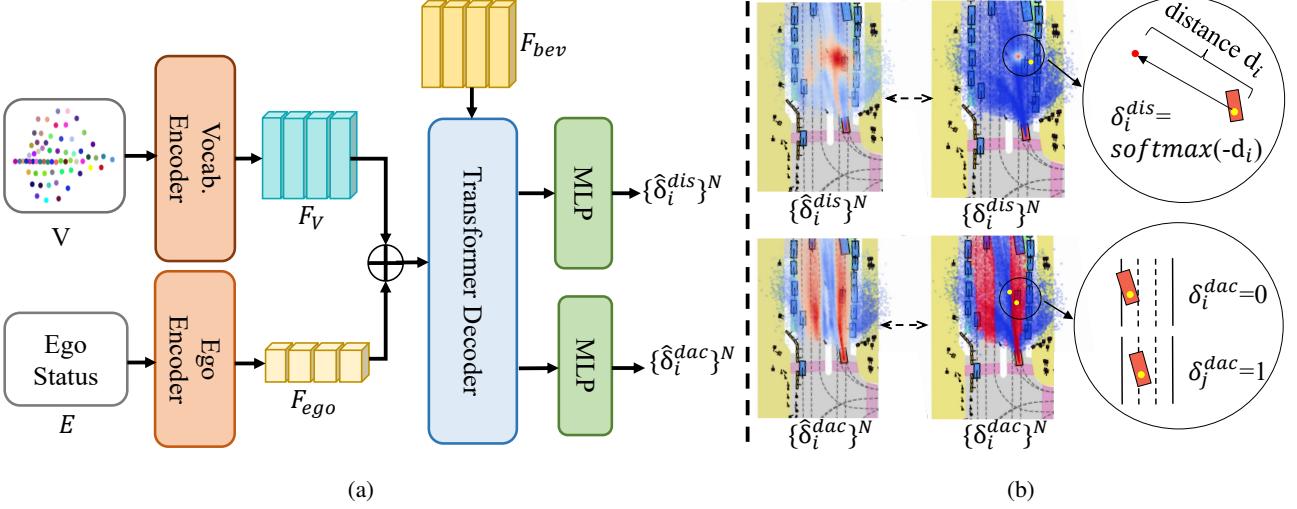


Figure 3. **Goal Point Scorer.** (a) shows the detailed structure of the Goal Point Construction Module, and (b) presents the score distributions of $\{\hat{\delta}_i^{dis}\}^N$, $\{\hat{\delta}_i^{dac}\}^N$, and $\{\hat{\delta}_i^{final}\}^N$, where points with higher scores are highlighted with warmer color.

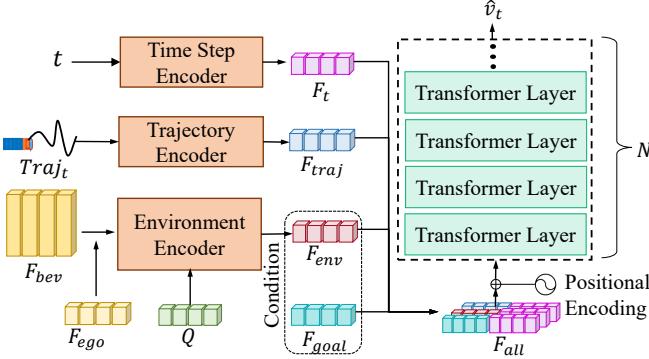


Figure 4. The network architecture used in Rectified Flow.

3.2.4. Trajectory Planning Module

In this module, we generate constrained, high-quality trajectory candidates using a generative model and then select the optimal trajectory through a scoring mechanism. Generative models based on diffusion methods like DDPM[14] and DDIM[26] typically require complex denoising paths, leading to significant time overhead during inference, which makes them unsuitable for real-time systems like autonomous driving. In contrast, Rectified Flow[24], which is based on the optimal transport path in flow matching, requires much fewer inference steps to achieve good results. We adopt Rectified Flow as the generative model, using the BEV feature and goal point as conditions to generate multimodal trajectories.

Multimodal Trajectories Generating. We generate multimodal trajectories by modeling the shift from the noise distribution to the target trajectory distribution. During this distribution transfer process, given the current state x_t and

time step t , we predict the shift \mathbf{v}_t .

$$\mathbf{v}_t = \tau^{norm} - x_0 \quad (4)$$

$$x_t = (1-t)x_0 + t\tau^{norm} \quad (5)$$

$$\tau^{norm} = \mathcal{H}(\tau^{gt}) \quad (6)$$

Where τ^{gt} is the ground truth trajectory and τ^{norm} is its normalized form. We define $\mathcal{H}(\cdot)$ as the normalization operation applied to the trajectory. The variable x_0 represents the noise distribution, which follows $x_0 \sim \mathcal{N}(0, \sigma^2 I)$. The variable x_t is obtained by linearly interpolating between x_0 and τ^{norm} .

As illustrated in Fig.4, we extract different features through a series of encoders. Specifically, we encode x_t using a linear layer, while t and the goal point are transformed into feature vectors via sinusoidal encoding. The feature F_{env} is obtained by passing the information from F_{bev} and F_{ego} through the environment encoder.

$$F_{env} = E_{env}(Q, (F_{BEV} + F_{ego}), (F_{BEV} + F_{ego})) \quad (7)$$

Here, E_{env} refers to a Transformer-based encoder, Q denotes a learnable embedding, and F_{ego} represents the ego status feature, which encodes the kinematic information of the ego vehicle.

We concatenate the features F_{env} , F_{goal} , F_{traj} , and F_t to form the overall feature F_{all} , which encapsulates the current state, time step, and scene information. This combined feature is then passed through several attention layers to predict the distribution shift \mathbf{v}_t .

$$\hat{\mathbf{v}}_t = \mathcal{G}(F_{all}, F_{all}, F_{all}) \quad (8)$$

$$F_{\text{all}} = \text{Concat}(F_{\text{env}}, F_{\text{goal}}, F_{\text{traj}}, F_t) \quad (9)$$

Where \mathcal{G} is the network that consists of N attention layers.

We reconstruct the trajectory distribution using x_0 and $\hat{\mathbf{v}}_t$. Typically, we achieve this by performing multiple inference steps through the Rectified Flow, gradually transforming the noise distribution x_0 to the target distribution τ^{norm} . Finally, we apply denormalization to τ^{norm} to obtain the final trajectory $\hat{\tau}$.

$$\hat{\tau} = \mathcal{H}^{-1}(\hat{\tau}^{\text{norm}}) \quad (10)$$

$$\hat{\tau}^{\text{norm}} = x_0 + \frac{1}{n} \sum_i^n \hat{v}_{t_i} \quad (11)$$

Where n is the total inference steps, and t_i is the time step sampled in the i -th step, which satisfies $t_i \in [0, 1]$. $\mathcal{H}^{-1}(\cdot)$ is the denormalization operation.

Trajectory Selecting In trajectory selection, methods like SparseDrive[27] and Diffusion-ES[32] rely on kinematic simulation of the generated trajectories to predict potential collisions with surrounding agents, thus selecting the optimal trajectory. This process significantly increases the inference time. We simplify this procedure by using the goal point as a reference for selecting the trajectory. Specifically, we trade off the trajectory distance to the goal point and ego progress, selecting the optimal trajectory through a trajectory scorer.

$$f(\hat{\tau}_i) = -\lambda_1 \Phi(f_{\text{dis}}(\hat{\tau}_i)) + \lambda_2 \Phi(f_{\text{pg}}(\hat{\tau}_i)) \quad (12)$$

where $\Phi(\cdot)$ is the minimax operation. $f_{\text{dis}}(\hat{\tau}_i)$ presents the L_2 distance of $\hat{\tau}_i$ and g , and $f_{\text{pg}}(\hat{\tau}_i)$ presents the L_2 distance of progress of $\hat{\tau}_i$ make.

Furthermore, predicted goal point may contain an error that can misguide the trajectory. To mitigate this, we mask the goal point during generation to create a shadow trajectory. If the shadow trajectory deviates significantly from the main trajectory, we treat the goal point as unreliable and use the shadow as the output.

3.2.5. Training Losses

Firstly, we optimize the perception extractor exclusively, and enforce multiple perception losses for supervision, including the cross-entropy loss for HD map (L_{HD}) and 3D bounding box classification (L_{bbox}) and L_1 loss for 3D bounding box locations (L_{loc}). This stage aims to enrich the BEV feature with information on various perceptions. Losses are as follows.

$$L_{\text{perception}} = w_1 * L_{HD} + w_2 * L_{bbox} + w_3 * L_{loc} \quad (13)$$

where w_1, w_2, w_3 are set to 10.0, 1.0, 10.0 in training. For the goal constructor, we employ the cross entropy loss for

distance score(L_{dis}) and DAC score(L_{dac}). w_4, w_5 are set to 1.0 and 0.005.

$$L_{\text{goal}} = w_4 * L_{\text{dis}} + w_5 * L_{\text{dac}} \quad (14)$$

$$L_{\text{dis}} = - \sum_{i=1}^N \delta_i^{\text{dis}} \log(\hat{\delta}_i^{\text{dis}}) \quad (15)$$

$$L_{\text{dac}} = -\delta^{\text{dac}} \log \hat{\delta}^{\text{dac}} - (1 - \delta^{\text{dac}}) \log(1 - \hat{\delta}^{\text{dac}}) \quad (16)$$

L_1 loss is utilized for multimodal planner.

$$L_{\text{planner}} = |\mathbf{v}_t - \hat{\mathbf{v}}_t| \quad (17)$$

4. Experiments

4.1. Dataset

Our experiment is validated on the OpenScene[6] dataset. OpenScene includes 120 hours of autonomous driving data. Its end-to-end environment Navsim[7] uses 1192 and 136 scenarios for trainval and testing, a total of over 10w samples at 2Hz. Each sample contains camera images from 8 perspectives, fused Lidar data from 5 sensors, ego status, and annotations for the map and objects.

4.2. Metrics

In the Navsim environment, the generated 2Hz, 4-second trajectories are interpolated via an LQR controller to yield 10Hz, 4-second trajectories. These trajectories are scored using closed-loop metrics, including No at-fault Collisions S_{NC} , Drivable Area Compliance S_{DAC} , Time to Collision S_{TTC} with bounds, Ego Progress S_{EP} , Comfort S_{CF} , and Driving Direction Compliance S_{DDC} . The final score is derived by aggregating these metrics. Due to practical constraints, S_{DDC} is omitted from the calculation¹.

$$S_{PDM} = S_{NC} \times S_{DAC} \times S_{TTC} \times \left(\frac{5 \times S_{EP} + 5 \times S_{CF} + 2 \times S_{DDC}}{12} \right) \quad (18)$$

4.3. Baselines

In Navsim, we compare against the following baselines: **Constant Velocity** Assumes constant speed from the current timestamp for forward movement. **Ego Status MLP** Takes only the current state as input and uses an MLP to generate the trajectory. **PDM-Closed** Using ground-truth perception as input, several trajectories are generated through a rule-based IDM method. The PDM scorer then selects the optimal trajectory from these as the output. **Transfuser** Uses both image and LiDAR inputs, fusing them via a transformer into a BEV feature, which is then used for trajectory generation. **LTF** A streamlined version

¹<https://github.com/autonomousvision/navsim/issues/14>

Method	Ego Stat.	Image	LiDAR	Video	$S_{NC} \uparrow$	$S_{DAC} \uparrow$	$S_{TTC} \uparrow$	$S_{CF} \uparrow$	$S_{EP} \uparrow$	$S_{PDM} \uparrow$
Constant Velocity	✓				68.0	57.8	50.0	100	19.4	20.6
Ego Status MLP	✓				93.0	77.3	83.6	100	62.8	65.6
LTF [3]	✓	✓			97.4	92.8	92.4	100	79.0	83.8
TransFuser [3]	✓	✓	✓		97.7	92.8	92.8	100	79.2	84.0
UniAD [15]	✓	✓	✓		97.8	91.9	92.9	100	78.8	83.4
PARA-Drive [30]	✓	✓	✓	✓	97.9	92.4	93.0	99.8	79.3	84.0
GoalFlow (Ours)	✓	✓	✓		98.4	98.3	94.6	100	85.0	90.3
GoalFlow [†]	✓	✓	✓		99.8	97.9	98.6	100	85.4	92.1
Human [‡]					100	100	100	99.9	87.5	94.8

Table 1. Comparisons with SOTA methods in PDM score metrics on Navsim [7] Test. Our method outperforms other approaches across all evaluation metrics. [†] uses the endpoint of the ground-truth trajectory as the goal point. [‡] uses the ground-truth trajectories to evaluate.

Model	Description	$S_{NC} \uparrow$	$S_{DAC} \uparrow$	$S_{TTC} \uparrow$	S_{CF}	$S_{EP} \uparrow$	$S_{PDM} \uparrow$
—	Transfuser[3]	97.7	92.8	92.8	100	79.0	84.0
\mathcal{M}_0	Base Model	97.9	94.2	94.2	100	79.9	85.6
\mathcal{M}_1	\mathcal{M}_0 + Distance Score Map	98.5	96.4	94.9	100	83.0	88.5
\mathcal{M}_2	\mathcal{M}_1 + DAC Score Map	98.6	97.5	94.7	100	83.8	89.4
\mathcal{M}_3	\mathcal{M}_2 + Trajectory Scorer	98.4	98.3	94.6	100	85.0	90.3

Table 2. Ablation study on the influence of each component. \mathcal{M}_0 is the base model, which uses rectified flow without goal point guidance and averages all generated trajectories to produce the final output. \mathcal{M}_1 and \mathcal{M}_2 introduce the distance score map and DAC score map, respectively, to guide the rectified flow. \mathcal{M}_3 builds upon \mathcal{M}_1 by incorporating trajectory scorer.

T	Inf. Time	$S_{NC} \uparrow$	$S_{DAC} \uparrow$	$S_{TTC} \uparrow$	$S_{CF} \uparrow$	$S_{EP} \uparrow$	$S_{PDM} \uparrow$
20	177.8ms	98.3	98.1	94.3	100	84.7	89.9
10	92.4ms	98.3	98.2	94.4	100	84.9	90.1
5	49.0ms	98.4	98.3	94.6	100	84.4	90.3
1	10.4ms	98.4	97.8	94.1	100	84.5	88.9

Table 3. Impact of different timesteps in inference. T denotes the number of denoising steps during inference. The results indicate that the model’s performance is robust to variations of denoising steps.

σ	$S_{NC} \uparrow$	$S_{DAC} \uparrow$	$S_{TTC} \uparrow$	$S_{CF} \uparrow$	$S_{EP} \uparrow$	$S_{PDM} \uparrow$
0.05	98.3	98.2	94.4	100	85.0	90.1
0.1	98.4	98.3	94.6	100	85.0	90.3
0.2	87.4	76.0	69.4	32.0	56.2	49.0
0.3	68.3	48.1	44.8	2.23	23.6	18.8

Table 4. Impact of different values of σ on the initial noise distribution. σ is the standard deviation of x_0 . The results show that performance drops significantly when σ exceeds 0.1, but remains stable for values below 0.1.

of Transfuser, where the LiDAR backbone is replaced with a learnable embedding. It achieves results in NavSim similar to Transfuser. **UniAD** Employs multiple transformer architectures to process information differently, using queries

to transfer information specifically for planning. **PARA-Drive** Differs from UniAD by performing mapping, planning, motion prediction, and occupancy prediction tasks in parallel based on the BEV feature.

4.4. Model Setups and Parameters

The training of rectified flow[24] follows classifier-free guidance[13], where features within the conditioning set are randomly masked to bolster model robustness. The last point of the ground-truth trajectory is used to guide flow matching in trajectory generation during training. In testing, the goal point for trajectory generation is set by selecting the highest-scoring point from the goal point vocabulary. The sampling process employs a smoothing method in [9] that re-scales the timesteps non-linearly, instead of using uniform intervals. We generate 128/256 trajectories, from which the trajectory scorer identifies the optimal one. All training was conducted on 4 nodes, each equipped with 8 RTX 4090 or RTX 3090 GPUs.

4.5. Results and Analysis

Comparison with SOTA Methods. In Table 1, we compared our method with several state-of-the-art algorithms in end-to-end autonomous driving, highlighting the highest scores in bold. Testing in the Navsim environment revealed

that GoalFlow consistently outperformed other methods in overall scores. Notably, our method surpasses the second-best approach by 5.5 points in the DAC score and by 5.7 points in the EP score, indicating that GoalFlow provides stronger constraints on keeping the vehicle within drivable areas, thus enhancing the safety of autonomous driving systems. Additionally, GoalFlow enables faster driving speeds while ensuring safety. Further experiments, where we replaced the predicted goal point with the endpoint of the ground truth trajectory, resulted in a score of 92.1, which is very close to the human trajectory score of 94.8. This demonstrates the strong guiding capability of the goal point in autonomous driving.

Ablation Study on The Influence of Each Component. We conduct an ablation study of the influence of each component in Table 2. The \mathcal{M}_0 represents a model that generates trajectories using only the rectified flow. In our experiment results, the base \mathcal{M}_0 consistently outperforms baseline methods on Navsim, particularly excelling in DAC and TTC. This indicates that the base model, which is based on flow matching, has effectively learned interactions with map information and surrounding agents, demonstrating that the flow model alone possesses strong modeling capabilities.

The \mathcal{M}_1 model builds on \mathcal{M}_0 by modeling the distance score distribution and selecting the point with the highest score to guide the rectified flow. We found that this results in the most significant improvement, demonstrating the effectiveness of decomposing the trajectory planning task. Specifically, we decompose the complex task into two simpler sub-tasks: goal point prediction and trajectory generation guided by the goal point.

The \mathcal{M}_2 model builds upon \mathcal{M}_1 by incorporating the prediction of DAC score distribution. The main improvement is seen in the DAC score. By introducing multiple evaluators from different perspectives, the model benefits from a more robust assessment, resulting in improved performance.

By incorporating trajectory scorer, which includes a trajectory selection and goal point checking mechanism, \mathcal{M}_3 further enhances the reliability of GoalFlow.

Impact of Different Steps in Inference. We conducted experiments with different denoising steps during the inference process, as shown in Table 3. In these experiments, We found as the number of inference steps decreases from 20 to 1, the scores remained stable. Specifically, even with just a single inference step, excellent performance was achieved. This highlights the advantage of flow matching over diffusion-based frameworks: flow matching takes a direct, straight path, requiring fewer steps to transfer from noisy distribution to target distribution during inference. Additionally, as the inference steps are reduced from 20 to 1, the denoising time in inference of one sample decreases

to 6% of the original. This efficient inference process is especially critical for autonomous driving systems, where real-time performance is essential.

Impact of Different Initial Noise in Training. In the experiments, the initial noise follows a Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$. We explored the impact of the noise variance on the generated trajectories in Table 4. The results reveal that noise settings have a significant impact on the scores. When the noise is set too high, the generated trajectories become excessively erratic; notably, with a σ of 0.3, the Comfort score drops to only 2.23, indicating that the trajectory lacks coherent shape. Conversely, when the noise variance is too low, flow matching tends to degenerate into a regression model, reducing the trajectory diversity available for scoring. This lack of variety lowers overall scores.

Dim	Backbone	S _N C ↑	S _D AC ↑	S _T TTC ↑	S _E P ↑	S _P DM ↑
256/256	V2-99/V2-99	97.1	96.2	91.8	81.8	86.5
512/512	V2-99/V2-99	97.3	97.6	92.5	83.0	88.1
1024/1024	V2-99/V2-99	98.6	97.5	94.7	85.0	89.4
256/256	resnet34/resnet34	98.3	93.8	94.3	79.8	85.7
1024/256	V2-99/resnet34	98.2	96.4	93.8	82.6	87.9

Table 5. **Impact of Scaling Model.** We examine the impact of scaling the Transformer’s hidden dimension and changing the image backbone within the *Goal Point Construction Module* (left) and *Trajectory Planning Module* (right). Increasing the hidden dimension and using a stronger image backbone both lead to improved end-to-end performance. For fair comparison, we align post-processing with baseline \mathcal{M}_2 .

Impact of Scaling Model. Inspired by [36], we present experiments on scaling the model based on the \mathcal{M}_2 in Table 5. Under the same V2-99 backbone, increasing the hidden dimension consistently improves performance, with the best results observed at a dimension of 1024. Additionally, we conducted experiments to compare different configurations of the Goal Point Construction Module. We found that scaling this module significantly improves overall performance, highlighting the critical role of goal point guidance in trajectory planning.

5. Conclusion

In this paper, we focus on generating accurate and efficient multimodal trajectories. We reviewed recent works on multimodal trajectory generation in autonomous driving and proposed a framework that generates precise goal points and effectively constrains the generative model with them, ultimately producing high-quality multimodal trajectories. We conducted experiments on the Navsim environment, demonstrating that GoalFlow achieves state-of-the-art performance. In the future, we aim to further investigate the impact of different guidance information on multimodal trajectory generation.

References

- [1] Shaoyu Chen, Bo Jiang, Hao Gao, Bencheng Liao, Qing Xu, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. *arXiv preprint arXiv:2402.13243*, 2024. [2](#), [4](#)
- [2] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. [3](#)
- [3] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *Pattern Analysis and Machine Intelligence (PAMI)*, 2023. [2](#), [4](#), [7](#)
- [4] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, page 1–9. IEEE Press, 2018. [2](#)
- [5] Felipe Codevilla, Eder Santana, Antonio Lopez, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9328–9337, 2019. [2](#)
- [6] OpenScene Contributors. Openscene: The largest up-to-date 3d occupancy prediction benchmark in autonomous driving. <https://github.com/OpenDriveLab/OpenScene>, 2023. [6](#)
- [7] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *arXiv*, 2406.15349, 2024. [1](#), [6](#), [7](#)
- [8] Patrick Dendorfer, Aljoša Ošep, and Laura Leal-Taixé. Goalgan: Multimodal trajectory prediction based on goal position estimation. In *Asian Conference on Computer Vision*, 2020. [2](#)
- [9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, pages 12606–12633. PMLR, 2024. [7](#)
- [10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press. [2](#)
- [11] Junru Gu, Chen Sun, and Hang Zhao. Densentnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15303–15312, 2021. [4](#)
- [12] Songen Gu, Wei Yin, Bu Jin, Xiaoyang Guo, Junming Wang, Haodong Li, Qian Zhang, and Xiaoxiao Long. Dome: Tampering diffusion model into high-fidelity controllable occupancy world model. *arXiv preprint arXiv:2410.10429*, 2024. [2](#)
- [13] Jonathan Ho. Classifier-free diffusion guidance. *ArXiv*, abs/2207.12598, 2022. [3](#), [7](#)
- [14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851. Curran Associates, Inc., 2020. [3](#), [5](#)
- [15] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. [1](#), [2](#), [7](#)
- [16] Zhiyu Huang, Xinshuo Weng, Maximilian Igl, Yuxiao Chen, Yulong Cao, Boris Ivanovic, Marco Pavone, and Chen Lv. Gen-drive: Enhancing diffusion generative driving policies with reward modeling and reinforcement learning fine-tuning. *arXiv preprint arXiv:2410.05582*, 2024. [1](#), [3](#)
- [17] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. *ICCV*, 2023. [1](#), [2](#), [3](#)
- [18] Chiyu “Max” Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, and Dragomir Anguelov. Motion-diffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9644–9653, 2023. [2](#), [3](#), [4](#)
- [19] Bu Jin, Xinyu Liu, Yupeng Zheng, Pengfei Li, Hao Zhao, Tong Zhang, Yuhang Zheng, Guyue Zhou, and Jingjing Liu. Adapt: Action-aware driving caption transformer. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7554–7561, 2023. [2](#)
- [20] Bu Jin, Yupeng Zheng, Pengfei Li, Weize Li, Yuhang Zheng, Sujie Hu, Xinyu Liu, Jinwei Zhu, Zhijie Yan, Haiyang Sun, Kun Zhan, Peng Jia, Xiaoxiao Long, Yilun Chen, and Hao Zhao. Tod3cap: Towards 3d dense captioning in outdoor scenes. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29 – October 4, 2024, Proceedings, Part XVIII*, page 367–384, Berlin, Heidelberg, 2024. Springer-Verlag. [2](#)
- [21] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. [2](#)
- [22] Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, et al. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024. [2](#)
- [23] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1–5, 2023. OpenReview.net*, 2023. [3](#)
- [24] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with

- rectified flow. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. 3, 5, 7
- [25] Robert J. McCann. A convexity principle for interacting gases. *Advances in Mathematics*, 128(1):153–179, 1997. 3
- [26] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, 2020. 3, 5
- [27] Wenchao Sun, Xuewu Lin, Yining Shi, Chuang Zhang, Haoran Wu, and Sifa Zheng. Sparsedrive: End-to-end autonomous driving via sparse scene representation. *arXiv preprint arXiv:2405.19620*, 2024. 1, 2, 3, 6
- [28] Junming Wang, Xingyu Zhang, Zebin Xing, Songen Gu, Xiaoyang Guo, Yang Hu, Ziying Song, Qian Zhang, Xiaoxiao Long, and Wei Yin. He-drive: Human-like end-to-end driving with vision language models. *arXiv preprint arXiv:2410.05051*, 2024. 2
- [29] Junming Wang, Xingyu Zhang, Zebin Xing, Songen Gu, Xiaoyang Guo, Yang Hu, Ziying Song, Qian Zhang, Xiaoxiao Long, and Wei Yin. He-drive: Human-like end-to-end driving with vision language models. *arXiv preprint arXiv:2410.05051*, 2024. 3
- [30] Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15449–15458, 2024. 2, 7
- [31] Zhou Xian, Nikolaos Gkanatsios, Theophile Gervet, Tsung-Wei Ke, and Katerina Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *Proceedings of The 7th Conference on Robot Learning*, pages 2323–2339. PMLR, 2023. 3
- [32] Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Jain, Jeff Schneider, and Katerina Fragkiadaki. Diffusion-es: Gradient-free planning with diffusion for autonomous driving and zero-shot instruction following. *arXiv preprint arXiv:2402.06559*, 2024. 2, 3, 4, 6
- [33] Tengju* Ye, Wei* Jing, Chunyong Hu, Shikun Huang, Lingsheng Gao, Fangzhen Li, Jingke Wang, Ke Guo, Wencong Xiao, Weibo Mao, Hang Zheng, Kun Li, Junbo Chen, and Kaicheng Yu. Fusionad: Multi-modality fusion for prediction and planning tasks of autonomous driving. 2023. *Equal Contribution. 2
- [34] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, Congcong Li, and Dragomir Anguelov. Tnt: Target-driven trajectory prediction. In *Proceedings of the 2020 Conference on Robot Learning*, pages 895–904. PMLR, 2021. 4
- [35] Wenzhao Zheng, Ruiqi Song, Xianda Guo, Chenming Zhang, and Long Chen. Genad: Generative end-to-end autonomous driving. *arXiv preprint arXiv: 2402.11502*, 2024. 2
- [36] Yupeng Zheng, Zhongpu Xia, Qichao Zhang, Teng Zhang, Ben Lu, Xiaochuang Huo, Chao Han, Yixian Li, Mengjie Yu, Bu Jin, Pengxuan Yang, Yuhang Zheng, Haifeng Yuan, Ke Jiang, Peng Jia, Xianpeng Lang, and Dongbin Zhao. Preliminary investigation into data scaling laws for imitation learning-based end-to-end autonomous driving. 2024. 8

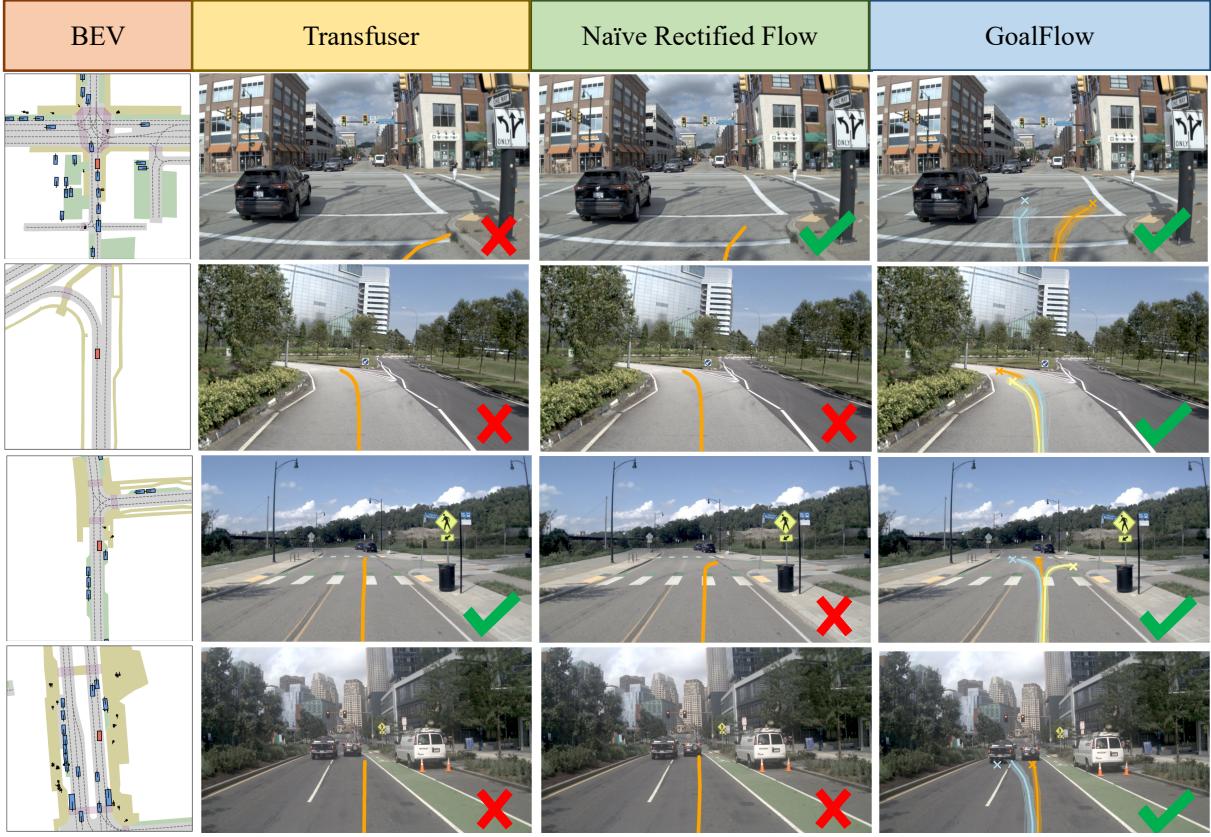


Figure 5. **Visualization of Trajectories.** \times indicates that the trajectory results in a collision or goes beyond the drivable area, while \checkmark represents a safe trajectory. The orange points are generated by the Goal Constructor, while the blue and yellow points correspond to samples from the vocabulary. The results highlight that GoalFlow generates higher-quality trajectories compared to the other two methods.

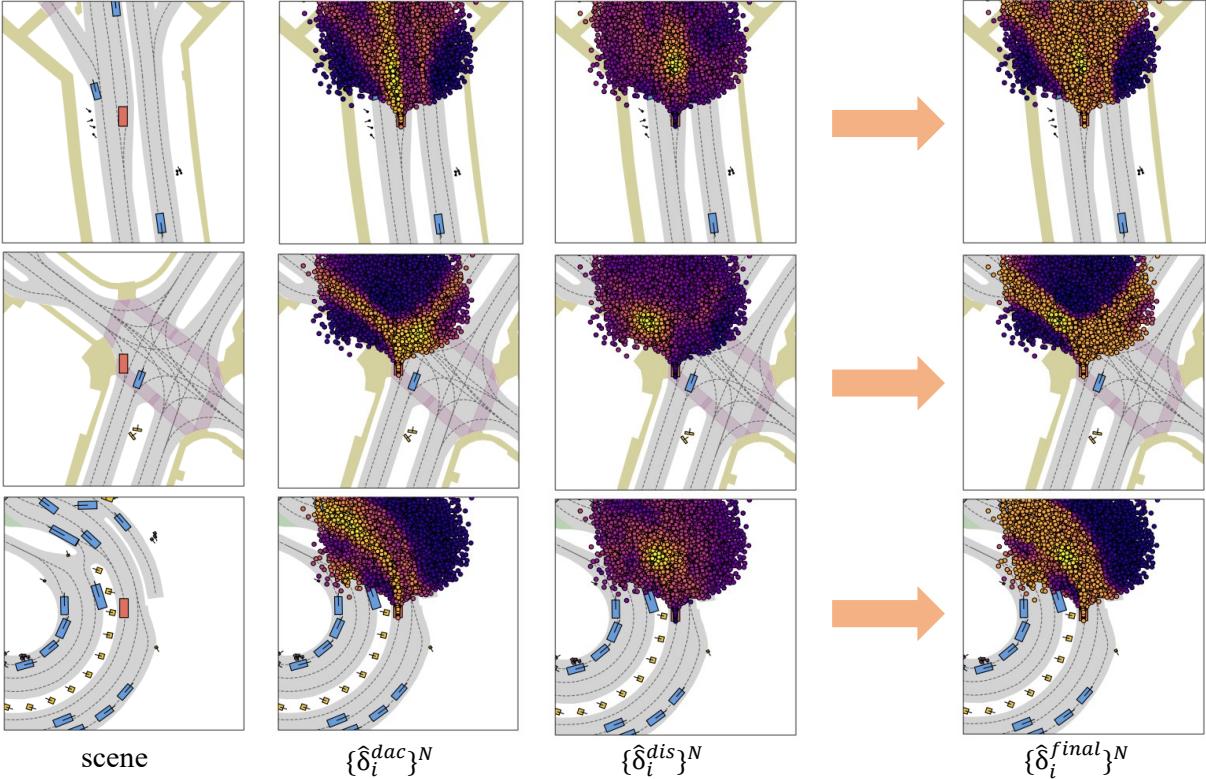


Figure 6. **Visualization of the goal point distribution.** The $\hat{\delta}_i^{dac}$ score indicates whether a point is within the drivable area, while the $\hat{\delta}_i^{dis}$ score reflects the distance relationship between the point and the goal. The final score $\hat{\delta}_i^{final}$ is a fusion of the $\hat{\delta}_i^{dac}$ and $\hat{\delta}_i^{dis}$ scores, where points with higher brightness represent higher scores.

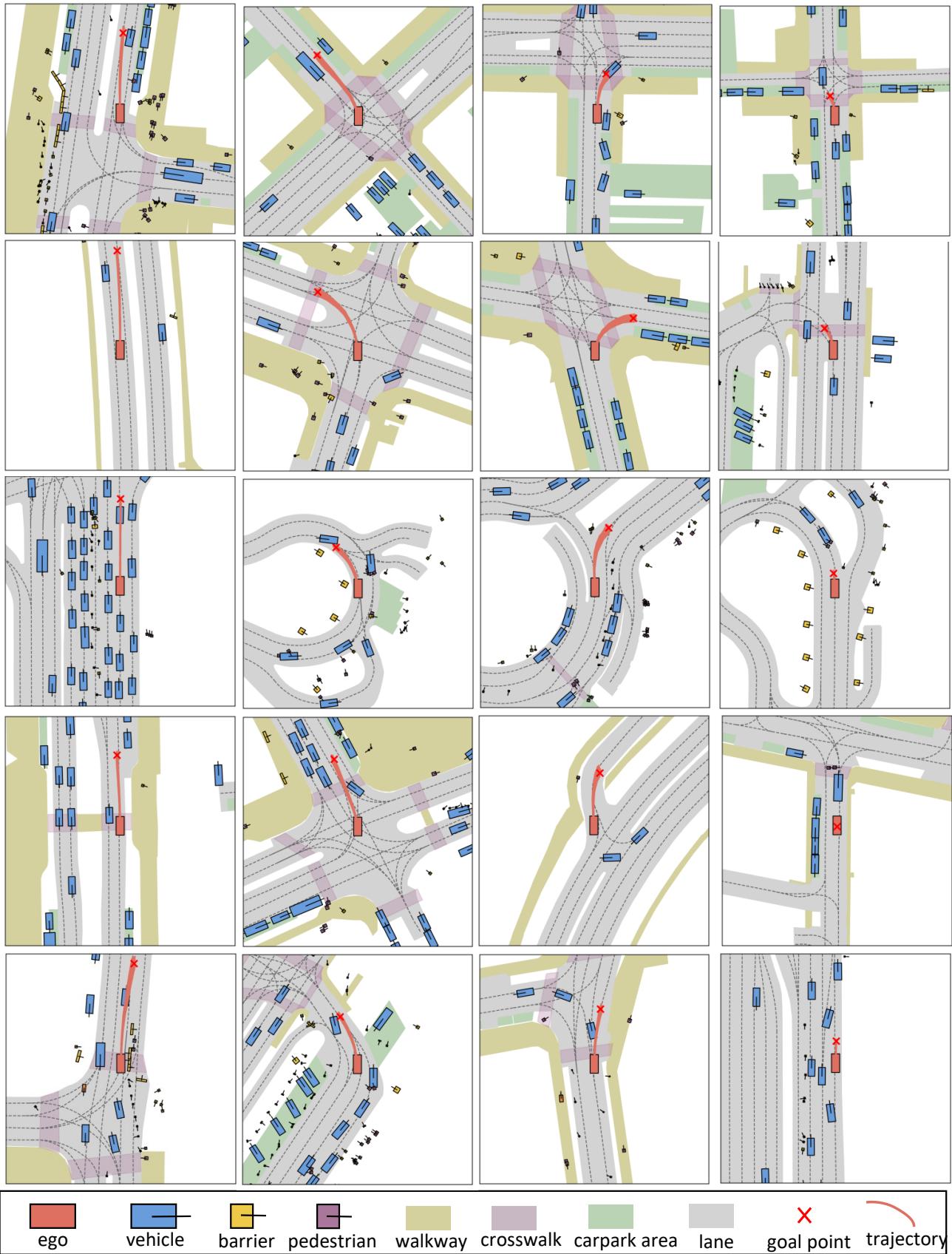


Figure 7. **Visualization of trajectories.** We visualize four scenarios: going straight, turning left, turning right, and yielding. For each scenario, 128 trajectories were generated using GoalFlow.