Learning Smooth Driving Policies via Bézier-Parameterized Flow Matching

Anonymous authors
Paper under double-blind review

Abstract

We propose a kinematically structured flow-matching planner for offline goal-conditioned behaviour cloning of driving data. Rather than directly predicting unconstrained trajectories in the ambient state-action space, the model parameterizes the action component of the learned vector field through low-order Bézier control points for heading and speed, then lifts these actions to form joint trajectories using a differentiable kinematic rollout. This induces a trajectory-level generative model whose samples remain aligned with the kinematics model used for execution, reducing lateral-velocity drift while maintaining smoothness without relying on post-hoc filtering. The model is trained with a flow-matching objective over full trajectories together with auxiliary supervision on fitted Bézier control points, enabling the use of a compact local motion representation while preserving conditional planning performance under scene and goal contexts. github:Code-tbd-on-acceptance.

1 Introduction

Learning autonomous driving policies directly from large-scale, real-world datasets has emerged as a powerful alternative to traditional, rule-based control systems (Treiber et al., 2000). Offline behaviour cloning (OBC) is particularly attractive in this paradigm, enabling the extraction of expert driving strategies from logs of human trajectories without the risks associated with online exploration. Motivated by the success of diffusion models (Ho et al., 2020; Song et al., 2021b; Lipman et al., 2023) and flow models (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Liu et al., 2023c) across diverse domains, e.g., (Dhariwal & Nichol, 2021; Karras et al., 2022), researchers have increasingly adopted diffusion architectures for OBC, e.g., (Jiang et al., 2024; Niedoba et al., 2023; Lu et al., 2025; Zheng et al., 2025; Tan et al., 2025), leveraging their powerful capacity to capture the nuanced, high-dimensional dynamics of human driving.

Despite the success of diffusion models within this domain, the highly structured nature of real-world driving trajectories presents a fundamental challenge. The data exhibits strong temporal correlations and is subject to strict physical constraints; however, standard diffusion models are agnostic to these underlying priors. By operating purely within an unstructured ambient data space, models are forced to learn these vital correlations and kinematics from noisy observations.

In this work we parameterize a diffusion planning model with a constrained hidden state that encodes tokenized Bézier control points for the heading and speed parameters of a kinematic bicycle model that are then used to recover vehicle states. This choice of parameterization provides reduction in dimension compared to directly planning in the ambient space while also providing a convenient method of constraining the smoothness of generated trajectories (e.g., jerk). We adopt an encoder-decoder based transformer model that tokenizes and learns scene feature embeddings that are then decoded into reduced order Bézier control point sequences that represent a trajectory action sequence. These control points are then integrated using a differentiable bicycle model to produce states in the original space.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 Related Work

Diffusion-based Planning Diffusion-based planners, and likewise flow-based planners, differ primarily in what object the generative model is defined over and where planning constraints enter the sampling procedure. Diffuser (Janner et al., 2022) models full state-action trajectories and performs planning by denoising trajectory samples, with task objectives incorporated through guidance during sampling rather than through a dynamics-constrained parameterization. Motion Planning Diffusion (Carvalho et al., 2024) similarly learns a diffusion prior over successful robot motion trajectories, but uses the model as a motion-planning prior for goal-conditioned posterior sampling, emphasizing multimodal trajectory generation and generalization to unseen obstacle configurations. Diffusion Policy (Chi et al., 2023) instead models short-horizon action sequences conditioned on observations and executes the receding-horizon action predictions in closed loop, which avoids explicit trajectory optimization but does not directly parameterize the state rollout. Conditional generative decision-making methods such as Decision Diffuser (Ajay et al., 2022) treat planning as conditional generation over trajectories, usually conditioning on returns, goals, or constraints, while diffusion policies for offline RL (Wang et al., 2023) use diffusion as an expressive policy class over actions.

Constrained Diffusion Models Various approaches have been proposed for adding constraints into diffusion planners by incorporating feasibility either by modifying the reverse process (Zhou et al., 2024; Naderiparizi et al., 2024; 2025) or by projecting/guiding samples toward admissible sets (Liu et al., 2023a; Feng et al., 2025), in similar ways to (Lou & Ermon, 2023; Fishman et al., 2023). SafeDiffuser (Xiao et al., 2025) augments denoising with control-barrier-function constraints, enforcing safety specifications during generation rather than changing the trajectory representation itself. Motion-planning diffusion methods commonly use learned priors together with posterior guidance, collision costs, or trajectory optimization to bias samples toward feasible motions (Carvalho et al., 2024). Flow-based planners, such as Flow Planner (Tan et al., 2025), use flow matching for autonomous driving, together with fine-grained trajectory tokenization and scene-conditioned interaction modelling. These flow formulations typically still predict trajectory velocities or trajectory tokens directly. Our approach is closest to trajectory-level diffusion and flow planners, but in contrast, our model defines the generative process over joint trajectories but parameterizes the learned flow field through a low-order action representation and a differentiable rollout map, so that smoothness enters through the vector-field architecture rather than only through guidance, filtering, or post-processing.

3 Problem Statement

We focus on an offline goal-conditioned behaviour cloning task without rewards. In particular, we assume all data is gathered from a reward-free Markov decision process $(\mathcal{S}, \mathcal{A}, p)$ where $\mathcal{S}$ and $\mathcal{A}$ are respectively the state and action spaces, augmented with a goal space $\mathcal{G} = \{\phi(s) \mid s \in \mathcal{S}\}$ where ϕ is a known state-to-goal mapping (e.g., wheel motion to absolute position), s.t., $p : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}_{\geq 0}$ is the state transition kernel where $s_{k+1} \sim p(\cdot \mid s_k, a_k, \phi(s_k))$. In this setting the agent cannot interact with the environment but only has access to a static dataset $\mathcal{D} = \{(\tau^{(i)}, c^{(i)}, g^{(i)})\}_{i=1}^{N}$ containing N samples of joint state-action trajectories, environment context, and goal conditions; where for i-th sample $\tau = (s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}) \in \mathcal{S}^H \times \mathcal{A}^H \subseteq \mathbb{R}^{H\times 2d}$ where $s_k \in \mathcal{S}$ denotes the state at horizon index k, and $a_k \in \mathcal{A}$ denotes the action applied at s_k, $c \in \mathcal{C}$ provides conditioning information about the action sequence a, and $g \in \mathcal{G}$ encodes goal conditioning. In what follows, we make the simplifying approximation that the dynamics are deterministic, and write $\tau = \mathcal{R}_f(s_0, a)$ to denote the rollout map w.r.t. the kinematic dynamics $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, s.t., given an initial state s_0 and action sequence $a = (a_0, \ldots, a_{H-1})$, the rollout is defined by $s_{k+1} = f(s_k, a_k)$ for $k = 0, \ldots, H-1$.

The objective is then to learn a conditional planner, of the form $\tilde{a}_{k:H} \sim p(\cdot \mid \tau_{0:k}, c, g)$, using $\tilde{\cdot}$ to distinguish model predictions from data which, given the current scene and goal information, generates a future action (or joint state-action) sequence whose rollout under the

transition dynamics model is aligned with the expert distribution while being kinematically smooth across the planning horizon.

4 Background

To tackle this we use a flow model (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Liu et al., 2023c), which is equivalent to diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b;a) under select parameterizations, following a similar approach to (Janner et al., 2022; Chi et al., 2023; Carvalho et al., 2024; Tan et al., 2025) where the forward process is defined over the entire trajectory horizon.

4.1 Flow Matching

Let $p_0(\tau \mid c, g)$ denote the conditional data distribution and $p_1(\tau)$ a limiting distribution, e.g., standard normal. Flow models learn a time-dependent vector field $v_\theta(\tau_t, c, g, t)$ which transports samples along the probability path induced by the chosen interpolation parameterization

$$\tau_t = \alpha_t \tau_0 + \beta_t \tau_1 \quad \text{s.t.} \quad \tau_0 \sim p_0(\cdot \mid c, g), \quad \tau_1 \sim p_1, \quad t \in [0, 1]. \tag{1}$$

Here $\alpha_t, \beta_t : [0, 1] \to [0, 1]$, s.t., $\alpha_0 = 1,\ \beta_0 = 0,\ \alpha_1 = 0,\ \beta_1 = 1$, are chosen to smoothly interpolate between the data and limiting distributions; e.g., the common choice is a linear interpolant with $\alpha_t = 1 - t$ and $\beta_t = t$. The conditioning variables (c, g) are fixed along the path. The conditional velocity along the path is derived as

$$\nu_t(\tau_t \mid \tau_0, \tau_1) = \frac{\partial}{\partial t}\alpha_t \tau_0 + \frac{\partial}{\partial t}\beta_t \tau_1, \tag{2}$$

and it is used to train velocity model v_θ under the following optimization task

$$\min_\theta \left\{ \mathbb{E}_{(\tau_0, c, g) \sim \mathcal{D},\ \tau_1 \sim p_1,\ t \sim \mathcal{U}([0,1])} \left[\| \nu_\psi(\tau_t, c, g, t) - \nu_t(\tau_t \mid \tau_0, \tau_1) \|_2^2 \right] \right\}. \tag{3}$$

Then at sampling time, a sample is obtained by drawing $\tau_1 \sim p_1$ and integrating the probability-flow backward from $t = 1$ to $t = 0$ using v_θ.

4.2 Bézier Curves

We make use of a structured latent space parameterized in terms of Bézier curves (Bézier, 1972) along with a kinematic model described in Section 5.1. The choice of Bézier curves is motivated by several reasons: providing smooth higher-order derivatives by construction, being efficient to evaluate using just one matrix multiplication, and providing endpoint tangent constraints to match velocity constraints during rollout.

Bézier curves are parameterized using a set of "control points," that jointly control the amount of curvature and coordinates of regions along the curve. To construct a Bézier curve, let $u = (u_0, \ldots, u_{H-1}) \in [0, 1]^H$ be a collocation vector with $0 = u_0 < u_1 < \cdots < u_{H-1} = 1$, which fixes the discrete evaluation points over the trajectory horizon, and denote the Bernstein basis (Bernstein, 1912) of order M by

$$B_r^M(u_k) = \binom{M}{r} (1 - u_k)^{M-r} u_k^r, \quad \text{for} \quad k = 0, \ldots, H-1, \quad r = 0, \ldots, M, \tag{4}$$

and corresponding matrix representation $B(u)^M \in \mathbb{R}^{H \times (M+1)}$ with entries $B(u)^M_{kr} = B_r^M(u_k)$. A Bézier curve is then evaluated as $B(u)^M z \in \mathbb{R}^{H \times d}$ for control points $z \in \mathbb{R}^{(M+1) \times d}$.

5 Proposed Method

In an attempt to improve the smoothness of generated trajectories and reduce the kinematic drift illustrated in Section 5.1, inspired by (Kondo et al., 2026), we parameterize the action component of the vector field using a low-order Bézier representation (Bézier, 1972) and then

lift this action representation to the full trajectory space through a differentiable kinematics rollout. This has the effect of regularizing actions, e.g., acceleration and turning, to be smooth, and due to rolling-out the predictions during training allows the model to self-correct for the kinematic drift illustrated in Figure 2.

5.1 Motivation

For our targeted planning application, dealing with recorded driving data, e.g., nuPlan (Caesar et al., 2021), directly parameterizing the vector field v_θ as an unconstrained vector in the joint state-action space of vehicle position, heading, and speed, $s = (x, y) \in \mathcal{S}$ and $a = (\theta, v) \in \mathcal{A}$, can produce trajectories that are not kinematically feasible, e.g., trajectories with discontinuous heading, oscillatory speed, or high jerk. To illustrate this, we can inspect the lateral velocity of observations w.r.t. the given heading angle and velocity derived from a one-step finite difference method between states; given consecutive positions s_k, s_{k+1}, time step $dt \geq 0$, and heading θ_k, define

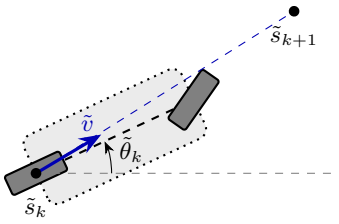


Figure 1: Single rollout step of the bicycle model in Eq. (6): $\tilde{s}_k$ is displaced along heading $\tilde{\theta}_k$ with speed $\tilde{v}$ to $\tilde{s}_{k+1}$.

$$(v_{k,x}, v_{k,y}) = \frac{s_{k+1} - s_k}{dt}, \quad v_{lat} = (v_{k,x}, v_{k,y}) \cdot (-\sin(\theta_k), \cos(\theta_k)). \tag{5}$$

As the kinematic model we opt to use a bicycle model (Matute et al., 2019; Scibior et al., 2021) and parameterize the steering angle δ in terms of instantaneous heading and speed; i.e., setting $\delta = \theta$. The discrete bicycle rollout is approximated by either initializing $\tilde{s}_0 = s_0$ and sequentially evaluating

$$\tilde{s}_{k+1} = \tilde{s}_k + \tilde{v}_k(\cos(\tilde{\theta}_k), \sin(\tilde{\theta}_k))dt, \quad \text{for} \quad k = 0, \ldots, H-1, \tag{6}$$

or if re-fitting at each step we sequentially reset $\tilde{s}_k = s_k$. Fitting this kinematic model to data and predictions is typically feasible with near zero[1] lateral velocity (slip) as the heading angle of the vehicle should be pointing predominantly in the direction of displacement; i.e., $\nu_{lat} \approx 0$. Fig. 2a shows that the distribution of lateral velocity for recorded trajectory data from the nuPlan dataset, see Appendix A, concentrates near this constraint. The trajectory rollouts from Flow Planner (Tan et al., 2025), our primary method of comparison that we base our architecture on, have noticeably increased variance. In contrast, our parameterization, which is detailed below, concentrates closer to the dataset lateral-velocity reflecting improved heading-alignment enforced by construction in our method. Fig. 2b provides a concrete example of how this misalignment can manifest into trajectory drift when plans are rolled-out under the kinematic driving model. Here we visualize the top-20 of 100 samples drawn from each method. These are re-integrated under the same bicycle-model update matching the evaluation of nuPlan, and plotted alongside the un-processed predictions. There is a noticeable mean shift between the Flow Planner predictions and what is executable, unlike our method which remains aligned with the prediction.

5.2 Model Parameterization

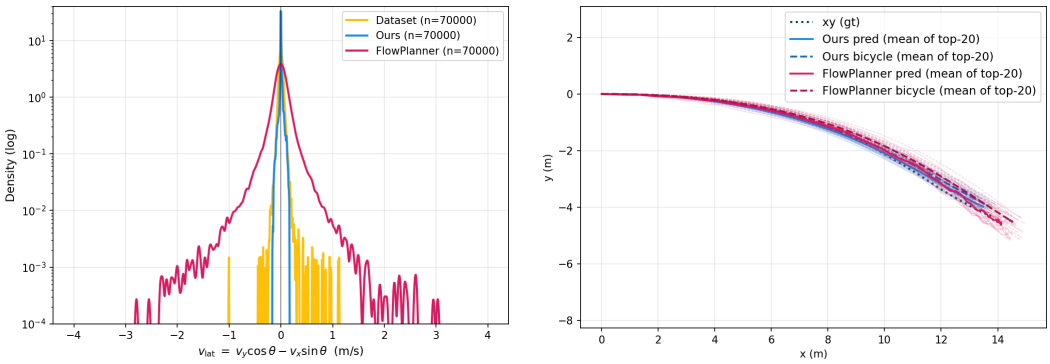
We re-parameterize $v_\theta(\cdot)$ using a goal conditioned action space model $z_\psi(\cdot)$ that predicts Bézier control points representing action sequences implemented using a modified version of the architecture in Tan et al. (2025); that is

$$a_\psi(\tau_t, c, g, t) = B(u)^M z_\psi(\tau_t, c, g, t) \in \mathbb{R}^{H \times d}, \tag{7}$$

s.t., given an initial state[2] s_0 and predicted action sequence $\tilde{a} = (\tilde{a}_0, \ldots, \tilde{a}_{H-1}) \sim a_\psi$, the model returns the rolled-out joint trajectory $\tilde{\tau} = \mathcal{R}_f(s_0, \tilde{a})$ w.r.t. the chosen kinematic model

[1] This value will not be exactly zero for real world data due to the presence of tire slip and sensor noise; whereas states exactly fit under a bicycle model will have zero lateral slip.

[2] For ego-centric data this is given as either the origin of the coordinate frame or a prior state.



(a) Distribution of lateral velocity for recorded trajectories and model rollouts. Under a kinematic bicycle, feasible motion satisfies $v_{lat} \approx 0$.

(b) Visualization of right-hand turn from nuPlan dataset with top-20 of 100 samples drawn from both Flow Planner and our proposed method.

Figure 2: Illustrations of systematic misalignment between baseline model predicted kinematics as compared to both the training dataset and our proposed method.

f; for compactness we write $\tau_\psi = \mathcal{R}_f(s_0, a_\psi)$. Notably, under this parameterization, u can be constructed to sample actions at adaptive rates above (or below) that of the discrete data samples used for training.

Thus the learned vector field is defined over the full trajectory space, while its action component is constrained by the chosen Bézier parameterization and the corresponding state component is determined by the kinematic rollout. The complete mapping is

$$z_\psi(\tau_t, c, g, t) \xrightarrow{B(u)} \tilde{a}_\psi(\tau_t, c, g, t) \xrightarrow{\mathcal{R}_f} \tilde{\tau}_\psi(\tau_t, c, g, t) \xrightarrow{\mathcal{V}_t} v_\theta(\tau_t, c, g, t). \tag{8}$$

Breaking down each stage of Eq. (8): Firstly, the predicted control points z_ψ define two Bézier curves of fixed order M, evaluated through $B(u)^M$ as in Eq. (7) to produce the action sequence $\tilde{a}_\psi$. The order M is selected in advance from the average dwell (halting) time in the dataset (see Appendix C.2; an adaptive alternative is discussed in Appendix C.3). Secondly, the rollout map $\mathcal{R}_f$ lifts the predicted actions to a predicted denoised trajectory by integrating them under the kinematic model, giving

$$\tilde{\tau}_\psi = (\tilde{s}, \tilde{a}), \quad \tilde{s}_{k+1} = f(\tilde{s}_k, \tilde{a}_k) \quad \text{for} \quad k = 0, \ldots, H-1. \tag{9}$$

Finally, the map $\mathcal{V}_t$ recovers the flow-matching velocity from the predicted denoised trajectory; under the linear path $\alpha_t = 1 - t$ and $\beta_t = t$ this is

$$\nu_\psi(\tau_t, c, w, g, t) = \frac{\tau_t - \tilde{\tau}_\psi(\tau_t, c, w, g, t)}{t}, \quad t > 0. \tag{10}$$

There are, however, additional subtleties in how the control points z_ψ and the basis evaluation $B(u)$ are parameterized in practice, namely trajectory chunking, tangent constraints, and warm starting, each of which is explained in the remainder of this section.

Trajectory Chunking To assist in tokenization and to break up planning into sub-actions, trajectories are chunked into multiple sub-trajectories, so that $\tau = \{\tau^{(j)}\}_{j=0}^{n}$ with $\tau^{(j)}_{h-w:h} = \tau^{(j+1)}_{0:w}$ for overlap window size $0 \le w < H$ and sub-horizon $0 < h < H$. The heading and speed actions are recovered respectively as

$$\tilde{a}_k^{(j,\theta)} = \sum_{r=0}^{M} B_r^M(u_k)\tilde{z}_r^{(j,\theta)}, \quad \tilde{a}_k^{(j,s)} = \sum_{r=0}^{M} B_r^M(u_k)\tilde{z}_r^{(j,s)} \tag{11}$$

with full action $\tilde{a}_k^{(j)} = (\tilde{a}_k^{(j,\theta)}, \tilde{a}_k^{(j,s)})$.

Algorithm 1 Training step

Input: Training dataset $\mathcal{D} = \{(\tau_0, c, g)\}$, Bernstein basis matrix $B^M(u)$, Rollout map $\mathcal{R}_f$, Reference distribution p_1, Learning rate η, condition drop probabilities p_c, p_w, p_g, Initialized networks z_ψ.
Output: Trained flow model v_ψ.

Sample $(\tau_0, c, g) \sim \mathcal{D}$
Sample $\tau_1 \sim p_1$
Sample $t \sim \mathcal{U}([\varepsilon, 1])$
$\tau_t \leftarrow (1-t)\tau_0 + t\tau_1$
$\nu_t \leftarrow \tau_1 - \tau_0$
Sample $c \leftarrow \bar{c} \sim p_c$
Sample $g \leftarrow \bar{g} \sim p_g$
Sample $w \sim \text{Bernoulli}(1 - p_w)$
▷ *Training step for action model*
$\tilde{z}_0 \leftarrow z_\psi(\tau_t, c, w, g, t)$
if $w = 1$ **then**
  $\tilde{z}_0^{(0,\theta)} \leftarrow \theta_0,\ \tilde{z}_0^{(0,s)} \leftarrow \nu_0$
end if
$\tilde{a}_0 \leftarrow B(u)^M \tilde{z}_0$
$\tilde{\tau}_0 \leftarrow \mathcal{R}_f(s_0, \tilde{a}_0)$
$\tilde{v} \leftarrow (\tau_t - \tilde{\tau}_0)/t$
$\mathcal{L}_{\text{FM}} \leftarrow \|\tilde{v} - \nu_t\|_2^2$
$\mathcal{L}_{\text{B}} \leftarrow \|\text{sg}(\hat{z}_0) - \tilde{z}_0\|_2^2$
$\mathcal{L}_{\text{S}} \leftarrow \|a_0 - \tilde{a}_0\|_2^2$
$L \leftarrow w_d\mathcal{L}_{\text{FM}} + w_b\mathcal{L}_{\text{B}} + w_s\mathcal{L}_{\text{S}}$
$\theta \leftarrow \theta - \eta\nabla_\theta L$

Algorithm 2 Sampling algorithm

Input: Context c, goal g, warm flag w and anchor $(\hat{\theta}, \hat{\nu})$, guidance weights λ_c, λ_g, trained network z_ψ, basis $B^M(u)$, rollout map $\mathcal{R}_f$, and reference distribution p_1; sampling steps K and step-size dt.
Output: Generated trajectory $\tilde{\tau}_0$.

Sample $\tau_1 \sim p_1$
Initialize $\tau \leftarrow \tau_1$
▷ *Sample trajectory*
for $k = K, \ldots, 1$ **do**
  $t \leftarrow k/K$
  $\tilde{z} \leftarrow z_\psi^\lambda(\tau, c, w, g, t)$
  if $w = 1$ **then**
    $\tilde{z}_0^{(0,\theta)} \leftarrow \hat{\theta},\ \tilde{z}_0^{(0,s)} \leftarrow \hat{\nu}$
  end if
  $\tilde{a} \leftarrow B(u)^M \tilde{z}$
  $\tilde{\tau}_0 \leftarrow \mathcal{R}_f(s_0, \tilde{a})$
  $\tilde{v} \leftarrow (\tau - \tilde{\tau}_0)/t$
  $\tau \leftarrow \tau - \tilde{v}\, dt$
end for
$\tilde{z} \leftarrow z_\psi^\lambda(\tau, c, w, g, t)$
if $w = 1$ **then**
  $\tilde{z}_0^{(0,\theta)} \leftarrow \hat{\theta},\ \tilde{z}_0^{(0,s)} \leftarrow \hat{\nu}$
end if
$\tilde{a} \leftarrow B(u)^M \tilde{z}$
$\tilde{\tau}_0 \leftarrow \mathcal{R}_f(s_0, \tilde{a})$
return $\tilde{\tau}_0$

Tangent Constraints To ensure smooth action transitions between neighbouring sub-trajectories, and to impose exact initial speed conditions at rollout, we impose tangent matching before evaluating the action sequence. For consecutive sub-trajectories j and $j+1$,

$$\hat{z}_0^{(j+1,\theta)} = \tilde{z}_M^{(j,\theta)}, \quad \hat{z}_1^{(j+1,\theta)} = 2\tilde{z}_M^{(j,\theta)} - \tilde{z}_{M-1}^{(j,\theta)}, \tag{12}$$

$$\hat{z}_0^{(j+1,s)} = \tilde{z}_M^{(j,s)}, \quad \hat{z}_1^{(j+1,s)} = 2\tilde{z}_M^{(j,s)} - \tilde{z}_{M-1}^{(j,s)}. \tag{13}$$

Warm Starting For the first control point in the first sub-trajectory we optionally override, before tangent constraints are applied, the initial speed prediction as

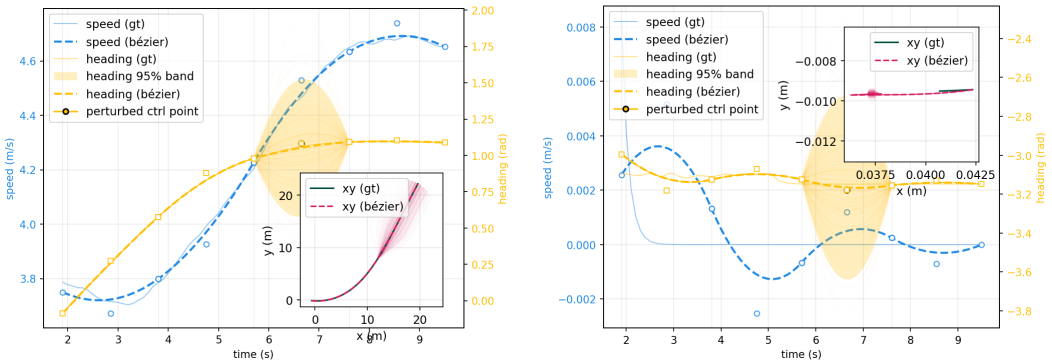
$$\tilde{z}_0^{(0,\theta)} = \theta, \quad \tilde{z}_0^{(0,s)} = \nu, \tag{14}$$

where $(\theta, \nu) = a_0$ during training, and for rollout step $i > 0$ to ensure subsequent predictions smoothly transition between auto-regressive samples; we refer to this as warm starting and use w to denote the flag in the model.

5.3 Training Objective

This model is trained using the flow-matching objective together with auxiliary supervision on self-consistency of predicted Bézier control points and speed regularization:

$$\min_\theta \Bigg\{ \mathbb{E}\Big[w_d \|v_\theta(\tau_t, c, w, g, t) - \nu_t(\tau_t \mid \tau_0, \tau_1)\|_2^2 + w_b \|\text{sg}(\hat{z}_0) - z_\psi(\tau_t, c, w, g, t)\|_2^2 + w_s \|a_0 - a_\psi(\tau_t, c, w, g, t)\|_2^2 \Big] \Bigg\}. \tag{15}$$



(a) nuPlan left-hand turn example.

(b) nuPlan (near) stationary example.

Figure 3: Illustration of Bézier features used to parameterize nuPlan data in terms of speed and heading of a kinematic bicycle model. Along each example a point controlling the heading is perturbed to show the localization of errors in prediction and the cumulative effect on the integration error.

with the expectation computed over $(\tau_0, c, g) \sim \mathcal{D}$, $\bar{c} \sim p_c$, $\bar{g} \sim p_g$, $w \sim Bernoulli(p_w)$, $t \sim \mathcal{U}(0,1)$, $\tau_t \sim q(\cdot|\tau_0)$. Here w_d, w_b, and w_s are hyperparameter weights for the flow-matching, Bézier control-point, and speed losses respectively. The term $\text{sg}(\hat{z}_0)$ is output from the model with tangent matching applied with stop-grad, the objective term serves to ensure self-consistency of predictions. See Appendix D.1 for an ablation of these loss terms.

5.4 Guidance Methods

We employ a training mixture for using classifier-free guidance following Ho & Salimans (2022); Harvey et al. (2022); Liu et al. (2023b); in particular, the conditions c, w, g are respectively modified to either drop a subset of nearest agents to the ego, a flag indicating warm start sampling where the initial speed predicted will be overwritten with the ground truth speed or prior prediction during rollouts, or dropping a subset of lanes from the map representing the driving corridor. The model is then sampled following

$$z_\psi^\lambda = (1 - \lambda_c - \lambda_g)\, z_\psi(\tau_t, \bar{c}, w, \bar{g}, t) + \lambda_c\, z_\psi(\tau_t, c, w, \bar{g}, t) + \lambda_g\, z_\psi(\tau_t, \bar{c}, w, g, t), \tag{16}$$

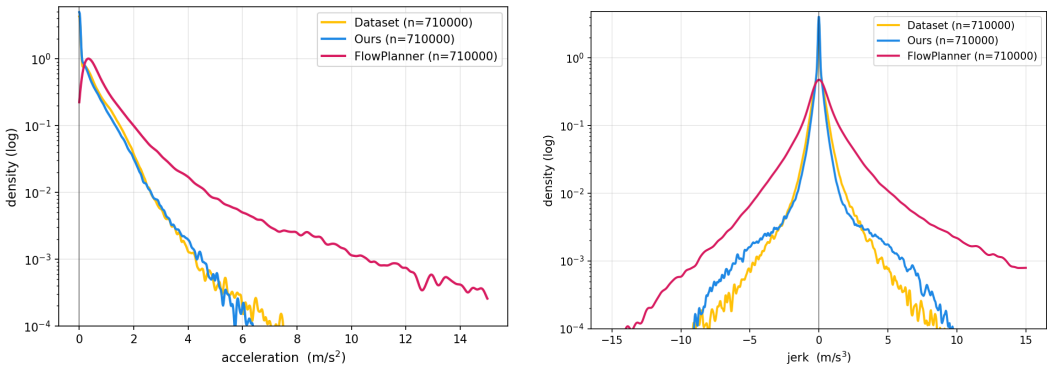
where $\bar{c}$ and $\bar{g}$ denote c and g with the k nearest agents and l corridor lanes removed, and λ_c, λ_g are the guidance weights. Additional details are discussed in Appendix C.4.

6 Experiments

In this work we focus on open-loop and closed-loop planning tasks for autonomous driving using the nuPlan dataset (Caesar et al., 2021). A detailed discussion on the data processing and filtering we perform over the dataset is given in Appendix A.

Bézier Features In order to illustrate how the Bézier basis described in Section 5 parameterizes trajectories we have produced a set of example curves from the nuPlan dataset in Fig. 3. In these examples we parameterize the speed, s, and heading angle, θ, of the kinematic bicycle model, and then integrate these curves to recover the states. As mentioned at the closing of Section 5.2 the choice of Bézier basis is motivated by improved locality of predictions over polynomial-like coefficients. As a demonstration, we perturb a random point along the heading curve of each sample and show the resulting effect on reconstruction.

Comfort Metrics and Smoothness Beyond strict kinematic feasibility, highlighted in Fig. 2b, in the context of driving, rider comfort is important. Various "comfort" metrics have been used for evaluating the smoothness and safety of driving trajectories; notably, bounds on maximum acceleration and jerk. Fig. 4 includes density plots for the acceleration and



(a) Distribution of per-step acceleration norm for recorded trajectories and model predictions.

(b) Distribution of per-step jerk for recorded trajectories and model predictions.

Figure 4: Kinematic smoothness of dataset trajectories compared to predictions made by Flow Planner and our proposed method. In both acceleration norm and jerk our proposed approach better matches the distribution of the expert driving behaviour of the dataset.

Model	Val14		Test14-hard		Test14	
	NR	R	NR	R	NR	R
PDM-Open* (Dauner et al., 2023)	53.53	54.24	33.51	35.83	52.81	57.23
GameFormer w/o refine. (Huang et al., 2023)	13.32	8.69	7.08	6.69	11.36	9.31
PlanTF (Cheng et al., 2023)	84.27	76.95	69.70	61.61	85.62	79.58
PLUTO w/o refine.* (Cheng et al., 2024)	88.89	78.11	70.03	59.74	89.90	78.62
Diffusion Planner (Zheng et al., 2025)	89.87	82.80	75.99	69.22	89.19	**82.93**
Flow Planner (Tan et al., 2025)	**90.43**	**83.31**	**76.47**	**70.42**	**89.88**	**82.93**
Flow Planner (Re-trained)	86.73	79.32	74.06	65.22	87.52	80.62
Bézier Planner (Ours)	85.68	75.26	72.11	59.44	85.76	79.29
+Warm	87.08	76.08	72.40	61.83	88.05	79.50
+Route*	87.71	77.78	73.08	61.83	86.76	80.42
+Warm & Route*	87.50	79.12	73.44	66.69	88.32	80.71

Table 1: Overall performance of learning-based planning baselines. Evaluation is conducted in both non-reactive (NR) and reactive (R) modes. An $*$ marks methods that directly leverage the road centerline from the map.

jerk of trajectories from the nuPlan dataset and our proposed method. For fair comparison between methods samples are filtered using the same Savitzky-Golay filter (Savitzky & Golay, 1964) and sliding window from nuPlan, see Appendix B for details, prior to computing any of the results. In terms of both acceleration norm and jerk our proposed approach better matches the distribution of the expert driving behaviour of the dataset in comparison to Flow Planner.

nuPlan Evaluation Performance The nuPlan evaluation provides an aggregate score of driving behaviour, measuring in part similar features discussed above in addition to avoidance of driving infractions under goals. We retrained Flow Planner to establish a baseline using the data-augmentation pipeline we use. We report the relative evaluation performance of our method against leading learning based methods on both the open-loop and closed-loop evaluation of nuPlan in Table 1. We report several variants of our method. The flag +Warm indicates the method was sampled using the warm start condition described in Section 5.2. The flag +Route indicates we filter g based on the expert trajectory creating a driving corridor of routes within $3m$ of the ego provided the ego traverses at least 10% of the arc length of the lane. This was implemented after observing a number of scenarios in the dataset have incorrectly filtered driving corridors (the route subset is provided as a feature in nuPlan) which are not in proximity to the ego.

Model	Collisions		TTC		Drivable		Comfort		Speed	
	NR	**R**	**NR**	**R**	**NR**	**R**	**NR**	**R**	**NR**	**R**
PDM-Open* (Dauner et al., 2023)	74.54	–	69.08	–	87.89	–	99.54	–	97.72	–
GameFormer w/o refine. (Huang et al., 2023)	–	–	–	–	–	–	–	–	–	–
PlanTF (Cheng et al., 2023)	94.13	–	90.73	–	96.79	–	93.67	–	97.78	–
PLUTO w/o refine.* (Cheng et al., 2024)	96.18	–	**93.28**	–	**98.53**	–	**96.41**	–	**98.13**	–
Diffusion Planner (Zheng et al., 2025)	–	–	–	–	–	–	–	–	–	–
Flow Planner (Tan et al., 2025)	–	–	–	–	–	–	–	–	–	–
Flow Planner (Re-trained)	95.66	93.38	91.23	**89.27**	93.83	94.54	93.74	86.05	97.51	**98.32**
Bézier Planner (Ours)	94.45	**95.14**	89.53	89.71	94.72	92.49	91.59	81.22	97.25	98.14
+Warm	95.75	94.41	90.70	89.27	95.62	94.28	93.39	85.51	97.33	98.20
+Route*	**96.20**	93.29	91.23	88.55	95.71	94.63	92.93	85.42	97.23	98.07
+Warm & Route*	95.84	93.87	90.88	**89.27**	95.80	**95.71**	93.83	**86.67**	97.31	98.06

Table 2: Individual statistic performance of learning-based planning baselines on Val14. Evaluation is conducted in both non-reactive (NR) and reactive (R) modes. An $*$ marks methods that directly leverage the road centerline from the map.

7 Conclusion

In this work we present a method of parameterizing a trajectory planner with Bézier curves using an explicit kinematic model that supports warm start sampling through manipulation of the model prediction. This method maintains reasonable performance against established baselines while offering improved prediction alignment to kinematic rollouts without post-hoc filtering.

8 Limitations

The primary limitation of our method is that trajectories must be sequentially rolled out, and due to the tangent constraints imposed between abutting Bézier curves, detailed in Appendix C, this cannot easily be parallelized. Consequently, the model throughput is approximately $0.85\times$ that of our baseline Flow Planner. Another limitation is due to how we opted to train this model end-to-end in trajectory space rather than pre-training the encoder to better exploit the dimension reduction of the Bézier latent space.

AI use statement

In this work, we used generative AI tools for providing feedback on research experiments by producing summary statistics, implement alterations to neural network designs, support qualitative data analysis by writing data processing code. We have not used generative AI tools for generating synthetic data sets, helping to develop theoretical models or conceptual frameworks, formulating mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, clean and reformat dataset. We have reviewed all AI-assisted work. We verified all LLM generated code for correctness and re-implementing methods when needed, we checked all summary results against experimental values, and we performed our own literature review. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This work studies generative models for autonomous driving and is evaluated using the nuPlan benchmark. The experiments are conducted in an offline research setting, and the results should not be interpreted as evidence of safety or suitability for deployment in real-world autonomous vehicles. Although this paper is specifically targeted to improve the generated trajectories in terms of feasibility and comfort, generative policies still may produce unsafe or otherwise undesirable trajectories, and deployment would require additional validation, safety mechanisms, and testing beyond the scope of this work.

Reproducibility statement

The main paper describes the proposed model and training objective as well as pseudo code for training and sampling algorithms used. Appendix C provides the architecture details and Appendix D provides training details and hyperparameters. Appendices A and B provide the nuPlan data preprocessing and evaluation protocol Additionally, we will release the codebase upon acceptance to facilitate reproducibility of our results.

References

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL https://openreview.net/forum?id=SUnVLf3djD7.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Florent Bartoccioni, Elias Ramzi, Victor Besnier, Shashanka Venkataramanan, Tuan-Hung Vu, Yihong Xu, Loick Chambon, Spyros Gidaris, Serkan Odabas, David Hurych, Renaud Marlet, Alexandre Boulch, Mickael Chen, Éloi Zablocki, Andrei Bursuc, Eduardo Valle, and Matthieu Cord. Vavim and vavam: Autonomous driving through video generative modeling, 2025. URL https://arxiv.org/abs/2502.15672.

Sergei N. Bernstein. Démonstration du théorème de weierstrass fondée sur le calcul des probabilités. *Communications of the Kharkov Mathematical Society*, 13(1):1–2, 1912.

Pierre Bézier. *Numerical Control: Mathematics and Applications*. John Wiley & Sons, 1972.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *CVPR ADP3 workshop*, 2021.

Joao Carvalho, An T. Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models, 2024. URL https://arxiv.org/abs/2308.01557v2.

Jie Cheng, Yingbing Chen, Xiaodong Mei, Bowen Yang, Bo Li, and Ming Liu. Rethinking imitation-based planner for autonomous driving, 2023. URL https://arxiv.org/abs/2309.10443.

Jie Cheng, Yingbing Chen, and Qifeng Chen. Pluto: Pushing the limit of imitation learning-based planning for autonomous driving, 2024. URL https://arxiv.org/abs/2404.14327.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning (CoRL)*, 2023.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=AAWuCvzaVt.

Berthy Feng, Ricardo Baptista, and Katherine Bouman. Neural approximate mirror maps for constrained diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vgZDcUetWS.

Nic Fishman, Leo Klarner, Valentin De Bortoli, Emile Mathieu, and Michael John Hutchinson. Diffusion models for constrained domains. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=xuWTFQ4VGO.

William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos, 2022. URL https://arxiv.org/abs/2205.11495.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Zhiyu Huang, Haochen Liu, and Chen Lv. Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3903–3913, October 2023.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Chiyu Max Jiang, Yijing Bai, Andre Cornman, Christopher Davis, Xiukun Huang, Hong Jeon, Sakshum Kulshrestha, John Wheatley Lambert, Shuangyu Li, Xuanyu Zhou, Carlos Fuertes, Chang Yuan, Mingxing Tan, Yin Zhou, and Dragomir Anguelov. Scenediffuser: Efficient and controllable driving simulation initialization and rollout. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=a4qT29Levh.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022.

Kota Kondo, Yuwei Wu, Vijay Kumar, and Jonathan P. How. Mighty: Hermite spline-based efficient trajectory planning. *IEEE Robotics and Automation Letters*, pp. 1–8, 2026. doi: 10.1109/LRA.2026.3681187.

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*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Guan-Horng Liu, Tianrong Chen, Evangelos Theodorou, and Molei Tao. Mirror diffusion models for constrained and watermarked generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=XPWEtXzlLy.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models, 2023b. URL https://arxiv.org/abs/2206.01714.

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*, 2023c. URL https://openreview.net/forum?id=XVjTT1nw5z.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Aaron Lou and Stefano Ermon. Reflected diffusion models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22675–22701. PMLR, 2023. URL https://proceedings.mlr.press/v202/lou23a.html.

Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7BQkXXM8Fy.

Jose A. Matute, Mauricio Marcano, Sergio Diaz, and Joshue Perez. Experimental validation of a kinematic bicycle model predictive control with lateral acceleration consideration. *IFAC-PapersOnLine*, 52(8):289–294, 2019. URL https://www.sciencedirect.com/science/article/pii/S2405896319304185.

Saeid Naderiparizi, Xiaoxuan Liang, Setareh Cohan, Berend Zwartsenberg, and Frank Wood. Don't be so negative! score-based generative modeling with oracle-assisted guidance. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=H8pMSJwRD5.

Saeid Naderiparizi, Xiaoxuan Liang, Berend Zwartsenberg, and Frank Wood. Constrained generative modeling with manually bridged diffusion models. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. URL https://doi.org/10.1609/aaai.v39i18.34159.

Matthew Niedoba, Jonathan Lavington, Yunpeng Liu, Vasileios Lioutas, Justice Sefas, Xiaoxuan Liang, Dylan Green, Setareh Dabiri, Berend Zwartsenberg, Adam Scibior, and Frank Wood. A diffusion-model of joint interactive navigation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 55995–56011. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/aeeddfbab4e99763ebac9221732c80dd-Paper-Conference.pdf.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Evgeny Y. Remez. Sur une propriété des polynômes de Tchebycheff. *Communications de l'Institut des Sciences Mathématiques et Mécaniques de l'Université de Kharkoff*, 13:93–95, 1936.

Walter Rudin. *Real and Complex Analysis*. McGraw-Hill, 3 edition, 1987.

Abraham. Savitzky and M.J.E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964. URL https://doi.org/10.1021/ac60214a047.

Adam Scibior, Vasileios Lioutas, Daniele Reda, Peyman Bateni, and Frank Wood. Imagining the road ahead: Multi-agent trajectory prediction via differentiable simulation. 2021. URL https://arxiv.org/abs/2104.11212.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265. PMLR, 2015. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Tianyi Tan, Yinan Zheng, Ruiming Liang, Zexu Wang, Kexin Zheng, Jinliang Zheng, Jianxiong Li, Xianyuan Zhan, and Jingjing Liu. Flow matching-based autonomous driving planning with advanced interactive behavior modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=MQVyuWBfSH.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24261–24272. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/cba0a4ee5ccd02fda0fe3f9a3e7b89fe-Paper.pdf.

Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Phys. Rev. E*, 62:1805–1824, 2000. URL https://link.aps.org/doi/10.1103/PhysRevE.62.1805.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Wei Xiao, Johnson (Tsun-Hsuan) Wang, Chuang Gan, Ramin Hasani, Mathias Lechner, and Daniela Rus. Safediffuser: Safe planning with diffusion probabilistic models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 100391–100412, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f95606d8e870020085990d9650b4f2a1-Paper-Conference.pdf.

Yinan Zheng, Ruiming Liang, Kexin ZHENG, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Diffusion-based planning for autonomous driving with flexible guidance. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wM2sfVgMDH.

Linqi Zhou, Aaron Lou, Samar Khanna, and Stefano Ermon. Denoising diffusion bridge models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=FKksTayvGo.

A nuPlan Data

This section describes the various data processing steps performed on the nuPlan dataset used for training and evaluation.

For a sample $(\tau_0, g, c) \sim \mathcal{D}$, the ego trajectory is τ_0. The scene context is represented as $c = \{A, L, O\}$, where $A = \text{KNN}_{B_A}(\tau_{0,0})$ denotes the B_A-nearest neighbouring agents around the ego at trajectory time $k = 0$, $L = \{(p_{ij})_{j=1}^{\ell}\}_{i=1}^{B_L}$ denotes equidistant points sampled along lane-lines of the roads, and $O = \{o_i\}_{i=1}^{B_O}$ denotes static obstacles and traffic controls. The goal is obtained by taking a subset of lanes from L to form a driving corridor with $R \subset L \, s.t., \, |R| \leq B_R$.

Trajectories are chunked up into multiple sub-trajectories to assist in tokenization and to breakup planning into sub-actions, so that $\tau_0^{(i)} = \{\tau_0^{(i,j)}\}_{j=0}^{n}$ with $\tau_{h-w:h}^{(i,j)} = \tau_{0:w}^{(i,j+1)}$ for overlap window size $0 \leq w < H$ sub-horizon $0 < h < H$. Following this, the data sample is pre-processed such that c is padded with the past h' observations (2sec) and are jointly rotated and centred around $\tau_{0,0}$.

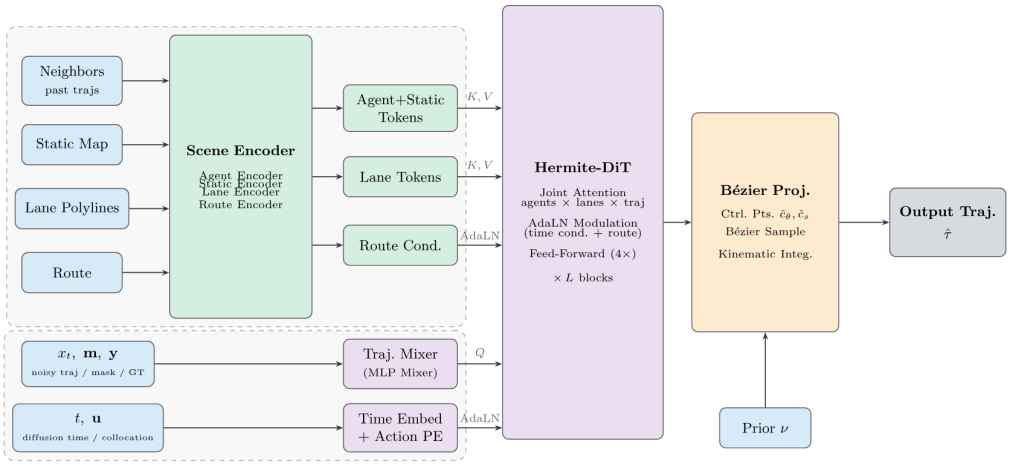


Figure 5: Diagram of proposed network architecture with hidden Bézier projection layer.

B nuPlan Metrics

The nuPlan development kit (`https://github.com/motional/nuplan-devkit`) contains established metrics, as mentioned in Section 6, for evaluating the smoothness of trajectory data. An important feature of the implementation of these metrics is both predicted (rolled-out) trajectories and ground truth version, used for the comparative metrics, are filtered using a Savitzky-Golay filter (Savitzky & Golay, 1964) with a sliding window prior to computing any of the metric results. Consequently, there is a subtle systematic misalignment between the executed trajectory within the metric computation and the rollout from the planner.

The measurements reported in Fig. 4a and Fig. 4b are computed following the same processes as in the nuPlan evaluation, however, we reduce the sliding window size for acceleration from $w = 8$ to $w = 5$ and for jerk from $w = 15$ to $w = 7$; this change corresponds to reducing the smoothing window from 1.5sec to 0.8sec. This was done to highlight the differences directly between predictions prior to the filtering performed during rollout.

C Network Architecture

We will now describe the parameterization of v_θ in greater detail along with the training and sampling algorithms and the inclusion of classifier-free guidance.

C.1 Network Design

A complete network diagram is included in Figure 5, the details of which we now describe.

Given a data sample $(\tau_0, g, c) \sim \mathcal{D}$, and diffusion step $\tau_t \sim q(\cdot|\tau_0)$. The scene features g, c are encoded following (Tan et al., 2025) using a sequence of MLP and MLPMixer blocks (Tolstikhin et al., 2021); the context and goal features as

$$c_{emb} = \text{MLP}(\text{pos} \oplus \text{MIXER}(\text{MLP}(\{A, L, O\}))) \in \mathbb{R}^{\{B_A, B_L, B_O\} \times e}, \tag{17}$$

$$g_{emb} = \text{MLP}(\text{pos} \oplus \text{MIXER}(\text{MLP}(R))) \in \mathbb{R}^{\{B_R\} \times e}, \tag{18}$$

$$\tag{19}$$

where $\oplus$ denotes vector concatenation and pos is a fixed sinusoidal positional encoding defined as

$$\text{pos}(p)_{2i} = \sin\left(\frac{p}{10000^{2i/e}}\right), \tag{20}$$

$$\text{pos}(p)_{2i+1} = \cos\left(\frac{p}{10000^{2i/e}}\right), \qquad i = 0, \ldots, \left\lfloor \frac{e}{2} \right\rfloor - 1 \tag{21}$$

for sequence position $p \in \{0, \dots, n-1\}$ and embedding dimension e.

The trajectory is then embedded and passed to the DiT decoder (Peebles & Xie, 2023) as

$$\tau_{emb} = \mathrm{MLP}(\mathrm{MIXER}(\tau_t)) \in \mathbb{R}^{n \times e}, \tag{22}$$

$$\tau_{token} = \mathrm{DiT}(\mathrm{pos}(t) \oplus \{\tau_{emb}, c_{emb}, g_{emb}\}) \in \mathbb{R}^{n \times e}, \tag{23}$$

overloading the positional embedding notation to represent the sinusoidal diffusion time embedding.

The decoded tokens are conditionally projected into Bézier control points for the speed and heading actions as

$$z^{(\theta)} = \mathrm{MLP}(\mathrm{pos}(0) \oplus \mathrm{MLP}(\tau_{token})) \in \mathbb{R}^{n \times (M+1)} \tag{24}$$

$$z^{(s)} = \mathrm{MLP}(\mathrm{pos}(1) \oplus \mathrm{MLP}(\tau_{token})) \in \mathbb{R}^{n \times (M+1)}. \tag{25}$$

As described in the vector field parameterization, these control points define the action sequence through multiplication by the Bernstein basis matrix. For sample i and sub-trajectory j, the heading and speed actions are recovered as

$$a_k^{(j,\theta)} = \sum_{r=0}^{M} B_r^M(u_k) z_r^{(j,\theta)}, \quad a_k^{(j,s)} = \sum_{r=0}^{M} B_r^M(u_k) z_r^{(j,s)} \tag{26}$$

with full action $a_k^{(j)} = (a_k^{(j,\theta)}, a_k^{(j,s)})$. To ensure smooth action transitions between neighbouring sub-trajectories, we impose tangent matching before evaluating the action sequence. For consecutive sub-trajectories j and $j+1$,

$$z_0^{(j+1,\theta)} = z_{M_{i,j}}^{(j,\theta)}, \quad z_1^{(j+1,\theta)} = 2z_{M_{ij}}^{(j,\theta)} - z_{M_{ij}-1}^{(j,\theta)}, \tag{27}$$

$$z_0^{(j+1,s)} = z_{M_{ij}}^{(j,s)}, \quad z_1^{(j+1,s)} = 2z_{M_{ij}}^{(j,s)} - z_{M_{ij}-1}^{(j,s)}. \tag{28}$$

The predicted states are then obtained by integrating the sampled actions with the kinematic model, giving

$$\tau_\psi = (s, a), \quad s_{k+1} = f(s_k, a_k) \quad \text{for} \quad k = 0, \dots, H-1. \tag{29}$$

The induced vector field $v_\theta(\tau_t, c, g, t)$ is then obtained from τ_ψ using the endpoint-to-velocity map described in the previous section.

C.2 Order Selection

Since we are representing each sub-trajectory using a Bézier curve of fixed order M it is desirable to have a rule of thumb for selecting a good starting value.

The limiting case is a dwell, in which the vehicle is stationary for part of a sub-trajectory and then moves; e.g., rolling stops. By the identity theorem (Rudin, 1987, Thm. 10.18), a polynomial exactly equal to zero on a sub-interval of $[0, 1]$ must be zero over the entire domain, so dwells taking a sufficient portion of an entire sub-trajectory forces $a^{(j,s)} \equiv 0$ over the whole sub-trajectory. To retain sufficient expressiveness to represent stop-start conditions between sub-trajectories, we instead require $|a^{(j,s)}(u)| \leq \varepsilon$ on the dwell interval $[0, \rho] \subset [0, 1]$ and $|a^{(j,s)}(1)| > 0$. By the extremal property of the Chebyshev polynomial T_M (Remez, 1936), this is attainable if and only if

$$M \geq \frac{\mathrm{arccosh}(|v|/\varepsilon)}{\mathrm{arccosh}(2/\rho - 1)}, \tag{30}$$

with $M-2$ in place of M for $j \geq 1$, where tangent matching constraints overwrite both $z_0^{(j,s)}$ and $z_1^{(j,s)}$. For a 1.9 sec sub-trajectory and $\varepsilon = 0.03\,\mathrm{m/s}$, $M = 8$ admits $\rho \leq 0.80$ representing 1.52 sec at $|v| = 5\,\mathrm{m/s}$ and $\rho \leq 0.73$ representing 1.39 s at $|v| = 15\,\mathrm{m/s}$. We opt to select $M = 8$.

C.3 Order Prediction

We initially attempted to implement a dynamic order prediction method in order to not over-parameterize each trajectory using a fixed M-order Bézier curve, which we suspected would increase the rollout sensitivity to errors in the prediction. This was implemented using a separate network $d_\omega(\cdot)$ which is trained in tandem to predict the best fit Bézier order, w.r.t. a pre-computed supervision target, for each trajectory. Namely, we suppose for $\tau_0 \sim D$ there exists a Bézier curve z of order $m \leq M$ s.t., $\|\tau_0 - \mathcal{R}_f(s_0, \tilde{a})\|_2^2 \leq \xi$ for $\tilde{a} = B(u)^m z$. We modified the model prediction from Eq. (7) as

$$m = d_\omega(c, g),$$
$$a_\psi(\tau_t, c, g, m, t) = B(u)^m z_\psi(\tau_t, c, g, m, t) \in \mathbb{R}^{H \times d};$$

notationally letting v_θ wrap the parameters of both the action and order networks. However, this resulted in worse rollout performance compared to using a fixed order so it was removed. We had additionally implemented a method of marginalizing over the predicted orders, but this required additional compute during training and only marginally improved performance.

C.4 Guidance Dropout and Sampling

In order to train the model using the classifier-free guidance mixture stated in Section 5.4 the neighbouring agents and route goals are sampled following:

$$A = \begin{cases} A & \text{with prob. } p_c \\ A \backslash \{n_i\}_{i=1}^{D_A} & \text{with prob. } 1 - p_c, \end{cases} \tag{31}$$

where $\{n_i\}_{i=1}^{D_A}$ denotes the set of $D_A \leq B_A$ closest agents and p_c are both hyperparameters during training, and

$$g = \begin{cases} R \subset L \, s.t., \, |R| \leq B_R & \text{with prob. } p_g \\ R = \emptyset & \text{with prob. } 1 - p_g, \end{cases} \tag{32}$$

where p_g is a hyperparameter during training.

The +Route flag indicates we filter R based on the expert trajectory creating a driving corridor of routes within $3m$ of the ego provided the ego traverses at least 10% of the arc length of the lane. This was implemented after observing a number of scenarios in the dataset have incorrectly filtered driving corridors (the route subset $R \subset L$ is provided as a feature in nuPlan) which are not in proximity to the ego.

The model is then sampled following

$$z_\psi^\lambda = (1 - \lambda_c - \lambda_g)\, z_\psi(\tau_t, \bar{c}, w, \bar{g}, t) + \lambda_c \, z_\psi(\tau_t, c, w, \bar{g}, t) + \lambda_g \, z_\psi(\tau_t, \bar{c}, w, g, t), \tag{33}$$

where $\bar{c}$ and $\bar{g}$ denote c and g with the D_A nearest agents and B_R corridor lanes removed, and λ_c, λ_g are the guidance weights. When $w = 1$ the first control point in the first sub-trajectory is set before tangent matching as

$$\tilde{z}_0^{(0,\theta)} = \theta, \quad \tilde{z}_0^{(0,s)} = \nu, \tag{34}$$

where $(\theta, \nu) = a_0$ during training, and for rollout step $i > 0$ to ensure subsequent predictions smoothly transition between auto-regressive samples; the first plan is sampled with $w = 0$.

D Experiment Details

All models were trained using either 4×NVIDIA L40S with gradient accumulation or 4×NVIDIA H100 (80GB) GPUs, using 1M training data samples from the nuPlan train data split following the re-processing described above in Appendix A. Models were trained for 400 epochs with an effective batch size of 2048. We use the AdamW (Loshchilov & Hutter, 2019) optimizer with a learning rate of 5×10^{-4} along with exponential moving average of model weights with an EMA decay of 0.999. Models were sampled using a second order centre weighted midpoint rule with 4 sampling steps. Network architecture parameters are detailed in Table 3. We use neighbouring agent dropout of $p_c = 0.3$ and goal dropout value of $p_g = 0.3$.

Type	Parameter	Symbol	Value
Training	Num. neighboring vehicles	B_A	32
	Num. past timestamps	h'	21
	Dim. neighboring vehicles	D_{neighbor}	11
	Num. lanes	B_L	70
	Num. points per polyline	l	20
	Dim. lanes	D_{lane}	12
	Num. navigation lanes	–	25
	Num. encoder block	–	3
	Num. decoder block	–	4
	Dim. encoder hidden layer	–	192
	Dim. decoder hidden layer	–	256
	Num. multi-head	–	8
	Len. future traj.	H	77
	Len. traj. segment	h	20
	Len. traj. overlap	w	1
Inference	Flow path	–	Affine
	Temperature	–	1.0
	Num. sampling steps	–	4

Table 3: Hyperparameters of Bézier planner evaluated in Table 1 and Table 2.

Model	Warm	No warm	Δ
Bézier Planner	76.23	78.68	
+Self-consistency	77.35	78.06	+1.12 / −0.62
Bézier Planner	86.79	86.66	
+Speed regularization	87.46	85.92	+0.67 / −0.74

Table 4: Effect of each training loss on the Val14 closed-loop score, non-reactive.

D.1 Training Objective Ablation

We ablate the different loss components included in Section 5.3 in Table 4.

D.2 Guidance Weight Ablation

The values reported in Table 1 and Table 2 were sampled using the classifier-free guidance method described in Section 5.4 using $\lambda_c = 1.8$ and $\lambda_g = 1.0$. These values were selected by running a parameter sweep over the nuPlan data split Test14 − hard for each model trained to 400 epochs and selecting the best configurations. The results of this sweep are reported in Table 5 and Table 6.

D.3 Flow Planner Baseline

In order to verify the data-processing performed, we re-trained Flow Planner, which serves as a primary point of comparison given its performance metrics, over the modified dataset. We follow the training procedure documented in (Tan et al., 2025) with the exception of reducing the planning horizon to match our method and re-parameterize the model output to directly produce heading angle.

λ_c	λ_g			
	0.6	1.0	1.8	2.6
0.6	69.09	68.45	69.49	70.52
1.0	70.25	69.61	69.57	70.19
1.8	71.46	72.11	70.71	70.06
2.6	70.77	72.10	70.52	70.21

Table 5: Bézier Planner: closed-loop score on Test14-hard (non-reactive, $n = 272$) over the guidance weights, λ_c on the neighbour branch and λ_g on the route branch. Warm start disabled, using control-point fusion.

λ_c	0.0	0.6	1.0	1.4	1.8	2.2	2.6
Score	61.26	68.53	72.29	74.06	73.63	72.94	73.07

Table 6: Flow Planner (re-trained): closed-loop score on Test14-hard (non-reactive, $n = 272$) over the guidance weight λ_c. This model carries no route branch, using trajectory fusion.

D.4 VaVAM Baseline

In order to demonstrate the proposed method applied to an additional architecture we adapt the Bézier formulation of the preceding sections to VaVAM (Bartoccioni et al., 2025), which is a first-person-view based driving policy built around a vision transformer. We train a baseline implementation of this method and our own Bézier variants from scratch over the nuPlan camera dataset. Performance results are reported in Table 7.

Model	**minADE@5 ↓**	**Comfort ↑**	
		origin	**no origin**
VaVAM-B (Bartoccioni et al., 2025)	0.530	–	–
VaVAM-B (our re-implementation)	0.469	0.084	0.107
VaVAM-B +Bézier	0.484	0.052	0.084
VaVAM-B +Bézier (u ×10)	0.478	0.050	0.080
VaVAM-B +Bézier (width ×2)	0.477	0.066	0.086
Ground truth (log replay)	–	0.729	0.740

Table 7: We adapt the Bézier formulation of the preceding sections to VaVAM, retraining VaVAM-B from scratch over the nuPlan dataset. The first learned row is our port of the authors' method; the remaining rows are Bézier variants. Both metrics are computed over nuPlan at $K = 5$. The (origin) column follows nuPlan eval, which anchors the boundary derivatives on the prepended ego-current state; (no origin) scores only predicted waypoints.