BC-Attack: Smooth Adversarial Trajectory Attack Based on Bézier Curves

Shuai Li[1], Libing Wu[2,*] Lijuan Huo[3,*] Zhuangzhuang Zhang[4], and Yi Yu[5]
School of Cyber Science and Engineering, Wuhan University, Wuhan, China
shuai_li@whu.edu.cn, wu@whu.edu.cn, lijuanHuo@whu.edu.cn,
zhzhuangzhuang@whu.edu.cn, yui1212@whu.edu.cn

***Abstract*—Trajectory prediction is a critical component for autonomous vehicles (AVs) to execute safe planning. While extensive work has been done on adversarial attacks against deep learning-based trajectory prediction models, most approaches generate adversarial trajectories by adding noise to trajectory points. This approach often results in unsmooth, or even jagged, trajectories, which suffer from poor stealthiness and are impractical for real-world implementation. To address this limitation, we propose a novel adversarial attack method based on Bézier Curves, named BC-Attack (Bézier Curves Attack). BC-Attack constructs adversarial trajectories by searching for control point positions. It leverages the continuous curvature of Bézier curves to ensure the generated adversarial trajectory naturally possesses geometric smoothness, while applying both soft and hard constraints to limit the offset error, velocity, and acceleration of the adversarial trajectory, improving its feasibility in the real world. Experiments on the Apolloscape and nuScenes datasets show that BC-Attack improves the trajectory prediction error by an average of 210% compared to normal prediction. Visualization of the trajectories also demonstrates that the generated adversarial trajectories are more stealthy and smoother than other attack methods.**

***Index Terms*—Autonomous Driving, Trajectory Prediction, Bézier Curve.**

I. Introduction

Autonomous driving is a complex system [1], [2] integrating modules such as environmental perception, behavior planning, and vehicle control. Trajectory prediction, as a crucial module between perception and planning, aims to predict the future trajectories of surrounding traffic participants based on their historical states, which is essential for the safe operation of autonomous vehicles. Therefore, numerous deep learning-based trajectory prediction methods [3]–[6], [19]–[28] have emerged in recent years, achieving promising results on general datasets. However, these data-driven models exhibit significant vulnerability to adversarial perturbations. This means that attackers can manipulate a vehicle to follow an adversarial trajectory, inducing the trajectory prediction model to output results that deviate drastically from its intended purpose. This could lead to dangerous decisions such as sudden braking or sharp turns by the autonomous vehicle. Therefore, in-depth research into the adversarial robustness of trajectory prediction models is of urgent security significance.

This work was supported by the National Natural Science Foundation of China (62441237, U24A20336, 62272348, U22B2022), and Wuhan City Joint Innovation Laboratory for Next-Generation Wireless Communication Industry Featuring Satellite-Terrestrial Integration under Grant 4050902040448.

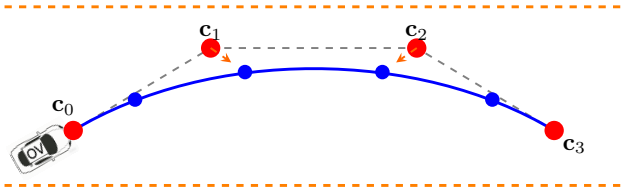


Fig. 1: Trajectory diagram based on Bézier Curve, $\mathbf{c}_0, \ldots, \mathbf{c}_3$ represents the control points of a Bézier Curve.

Existing trajectory prediction adversarial attack methods [7]–[14] primarily focus on addressing two core challenges. One is the effectiveness of the attack, namely, whether the attack can successfully interfere with the trajectory prediction model's prediction of the future intentions of surrounding vehicles, thereby affecting the autonomous vehicle's own future trajectory planning and leading to traffic accidents such as collisions. The other challenge is the physical realism of the adversarial trajectory, that is, the adversarial trajectory generated by the attack method needs to be smooth and continuous enough, and conform to the laws of vehicle kinematics (such as continuous speed and acceleration), so that it can be replicated in the real world by controlling the vehicle.

Nevertheless, very few works can solve both of these challenges simultaneously. The work [7], [8] that focus on the effectiveness of attack pays more attention to improving the prediction error of the trajectory prediction model while neglecting the smoothness of the trajectory. The resulting adversarial trajectories exhibit high-frequency jitter and jagged edges, making them easily detectable and physically unreproducible. At the same time, the works [9], [11]–[14] dedicated to practical deployment reconstruct or smooth the trajectory to satisfy kinematic constraints, achieving sufficient smoothness but sacrificing attack effectiveness. In short, existing methods cannot simultaneously solve both problems.

To simultaneously address both of the aforementioned issues, we propose a novel adversarial attack framework named BC-Attack (Bézier Curves Attack). Instead of generating adversarial trajectories through point perturbation optimization, BC-Attack innovatively uses Bézier Curves [33]–[35]. A Bézier Curve is defined by a set of control points, with

each point on the curve being a weighted combination of these control points, as shown in Fig. 1. These control points act like "magnets", guiding the curve's direction. Based on this characteristic, BC-Attack shifts the optimization objective from discrete trajectory points to control points. By adjusting the positions of these control points, the target Bézier Curve is obtained, and points are selected on the curve according to the time series of historical trajectories to generate the adversarial trajectory.

Since the Bézier Curve is a smooth polynomial parametric curve, the adversarial trajectories generated by BC-Attack inherit its smoothness, effectively *solving the problem of trajectory non-smoothness* in traditional point-by-point perturbation methods. Furthermore, the combination of physical constraints applied to the trajectory (velocity, acceleration, and control point amplitude pruning) makes *the adversarial trajectory generated by BC-Attack feasible in the real world*. Moreover, we conducted extensive evaluations of various state-of-the-art prediction models [4]–[6] on two mainstream autonomous driving datasets, Apolloscape [15] and nuScenes [16]. Experimental results show that BC-Attack significantly outperforms baseline methods in inducing prediction models to generate prediction errors. This indicates that the trajectory generated by BC-Attack is not only smooth but also has a significant deception effect, *effectively solving two challenges simultaneously*.

Our main contributions are summarized as follows:

- We propose a novel adversarial attack framework based on Bézier Curves. By optimizing control points to construct adversarial trajectories, we ensure that the generated adversarial trajectories possess naturally continuous curvature and smoothness.
- We designed a soft-hard constraint strategy, which applies soft constraints on velocity and acceleration and hard constraints on offset range to ensure the physical feasibility of the trajectory.
- We conducted comprehensive comparative evaluation experiments on three prediction models and two trajectory datasets. Our method achieves a 210% improvement in average prediction error compared to the normal case, and improvements of 2.50% and 0.43% compared to the two baseline attacks, respectively.

II. RELATED WORKS

A. Trajectory Prediction for Autonomous Driving

In autonomous driving systems, the core objective of trajectory prediction is to predict the future trajectories of surrounding traffic participants (such as vehicles and pedestrians) based on their historical states. The accuracy of this prediction is crucial for safe behavioral decisions, obstacle avoidance, and path planning by autonomous vehicles. In recent years, with the development of deep learning, numerous data-driven prediction models have emerged. For example, Trajectron++ [4] employs a spatiotemporal graph structure combined with recurrent neural networks (RNNs) to model the interactions between agents and their historical dynamics; GRIP [5] utilizes graph convolutional networks (GCNs) to efficiently capture spatiotemporal features in traffic scenarios; and FQA [6] proposes a novel fuzzy query attention mechanism combined with long short-term memory networks (LSTMs) to flexibly model interactions between agents through continuous-valued fuzzy decision-making. These models have demonstrated excellent prediction accuracy in benchmark tests, but the robustness of these trajectory prediction models against malicious interference remains an unresolved safety issue.

B. Adversarial Attacks on Trajectory Prediction

Adversarial attacks aim to mislead deep learning model outputs by introducing subtle perturbations to inputs. Current adversarial attacks targeting trajectory prediction primarily fall into two categories: indirect attacks and direct attacks. Indirect attacks [17], [18] generate adversarial point clouds by placing paper near stationary vehicles in the real world, disrupting autonomous driving perception [29], [30] and tracking modules [31]. This causes erroneous perception of the vehicle's bounding box attributes, leading to inaccurate position localization and indirectly generating adversarial trajectories to attack the prediction module. Direct attacks, conversely, introduce perturbations directly into the prediction model's input (i.e., historical trajectory data) to generate adversarial trajectories. While differing in implementation, both attack methods share the same objective: directly or indirectly compromising the prediction module through adversarial trajectories.

Current direct attacks face a compatibility challenge between attack effectiveness and trajectory realism. Early approaches like Zhang *et al.* [7]. optimized by adding noise directly to trajectory points. While this effectively induces prediction errors, it often generates jagged trajectories violating kinematic laws, making attacks difficult to reproduce in real-world scenarios. To address this, some studies [9]–[12] adopted a two-stage "attack-then-reconstruct" strategy: first optimizing trajectory points to generate highly effective adversarial trajectories, then reconstructing them using kinematic or continuous curvature models to ensure smoothness. However, this post-processing step smooths out carefully designed adversarial features, significantly reducing the attack effectiveness of the final generated trajectories.

III. PROBLEM FORMULATION

A. Trajectory Prediction Task

Trajectory Prediction forecasts the future trajectories of road agents (vehicles and pedestrians) based on their current and historical states perceived over T_{obs} time steps. Let there be N agents in the scene. The state of the i-th agent at time step t is denoted as $\mathbf{s}_t^i \in \mathbb{R}^D$, where D is the state dimension (e.g., coordinates, velocity, acceleration). We denote the historical trajectories as input tensor $\mathbf{X}_t = \{\mathbf{X}_t^i \mid i = 1, \ldots, N\}$, where $\mathbf{X}_t^i = (\mathbf{s}_{t-T_{obs}+1}^i, \ldots, \mathbf{s}_t^i)$ represents the sequence of states for agent i at time step t. The goal of the prediction model $f_\theta(\cdot)$ is to generate predicted trajectories $\mathbf{Y}_t = \{\mathbf{Y}_t^i \mid i = 1, \ldots, N\}$ for a future horizon of T_{pred} time steps given $\mathbf{X}_t$ and optional context $\mathbf{I}$, where $\mathbf{Y}_t^i = (\mathbf{s}_{t+1}^i, \ldots, \mathbf{s}_{t+T_{pred}}^i)$. In this work, we

focus on deep learning-based predictors trained to minimize the distance error between $\mathbf{Y}_t$ and the true future trajectory $\mathbf{F}_t = \{\mathbf{F}_t^i \mid i = 1, \ldots, N\}$, where $\mathbf{F}_t^i = (f_{t+1}^i, \ldots, f_{t+T_{pred}}^i)$.

B. Adversarial Attack Model

We consider a white-box attack scenario where the adversary controls a specific target vehicle v_{tar} to mislead the prediction model. The adversary aims to maximize the prediction error by adding a subtle perturbation $\boldsymbol{\delta}$ to the historical trajectory $\mathbf{X}_t^{tar}$, resulting in an adversarial trajectory $\tilde{\mathbf{X}}_t^{tar} = \mathbf{X}_t^{tar} + \boldsymbol{\delta} = (\hat{\mathbf{s}}_{t-T_{obs}+1}, \ldots, \hat{\mathbf{s}}_t)$. The attack is formulated as an optimization problem:

$$\max_{\boldsymbol{\delta}} \mathcal{L}_{attack} \left(f_\theta(\tilde{\mathbf{X}}_t^{tar}, \mathbf{I}), \mathbf{F}_t^{tar} \right) \tag{1}$$

where $\mathcal{L}_{attack}$ is the loss function that measures the error between the predicted trajectory $\mathbf{Y}_t^{tar}$ and the true future trajectory $\mathbf{F}_t^{tar}$.

IV. Methodology

A. Overview

To address the limitations of existing methods, we propose **BC-Attack**, a trajectory prediction adversarial attack framework based on Bézier Curves. The core idea of our method is to parameterize historical trajectories using the principles of Bézier Curves, optimize the positions of control points to obtain the target Bézier Curve, and then extract points on the curve according to the time series to obtain the adversarial trajectory. The attack process is described in Alg. 1. BC-Attack utilizes the mathematical properties of Bézier Curves to fundamentally guarantee the smoothness of the generated trajectory.

The attack process specifically includes two stages: (1) **Bézier Curve Parameterization**: precalculate the mapping matrix from trajectory points to control points, solve for the initial control points from the original trajectory using a pseudo-inverse method with regularization, and add random noise to find the optimal adversarial trajectory over a wider range, avoiding getting trapped in local optima; (2) **Adversarial Optimization**: under hard and soft constraints, iteratively update the control point perturbation to maximize the prediction error. We will elaborate on each stage below.

B. Bézier Curve Parameterization

Bézier Curves are parametric curves widely used in computer graphics and path planning, with the core idea of defining a smooth curve through a small number of control points. In this method, we leverage Bézier Curves to parameterize vehicle trajectories, shifting the optimization objective from discrete trajectory points to control points.

Specifically, an $n = K - 1$ degree Bézier Curve is defined by K control points $\mathbf{C} = [\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{K-1}]^\top \in \mathbb{R}^{K \times 2}$, where each control point $\mathbf{c}_i$ represents a coordinate in the two-dimensional plane. A point on the curve at parameter $\tau \in [0, 1]$ can be expressed as a weighted combination of the control points:

$$\mathbf{p}(\tau) = \sum_{i=0}^{K-1} B_{i,n}(\tau) \mathbf{c}_i \tag{2}$$

where $B_{i,n}(\tau) = \binom{n}{i} \tau^i (1-\tau)^{n-i}$ is the Bernstein basis polynomial, serving as the weight coefficient for each control point. As the parameter τ varies from 0 to 1, the curve smoothly transitions from the starting control point $\mathbf{c}_0$ to the ending control point $\mathbf{c}_{K-1}$, while the intermediate control points determine the curvature and direction of the curve through their "pulling" effect.

To generate an adversarial trajectory with T_{obs} discrete time steps, we normalize the time indices $t = 0, 1, \ldots, T_{obs} - 1$ to $\tau_t = t/(T_{obs} - 1)$, uniformly distributing them within the interval $[0, 1]$. Based on this, we construct the Bernstein basis matrix $\mathbf{M} \in \mathbb{R}^{T_{obs} \times K}$, where the element $M_{t,i} = B_{i,n}(\tau_t)$ represents the weight of the i-th control point at the t-th time step. Since the normalized parameters τ_t depend only on the number of time steps T_{obs} and control points K, the matrix $\mathbf{M}$ can be precomputed before optimization and remains constant throughout the process.

By precomputing the basis matrix $\mathbf{M}$, the trajectory generation process can be simplified to a single matrix multiplication:

$$\mathbf{X} = \mathbf{M}\mathbf{C} \tag{3}$$

where $\mathbf{X} \in \mathbb{R}^{T_{obs} \times 2}$ denotes the generated trajectory sequence.

Conversely, to parameterize the original trajectory, we need to solve for the control points $\mathbf{C}$ from Eq. 3 given the original historical trajectory $\mathbf{X}_t^{tar} \in \mathbb{R}^{T_{obs} \times 2}$. Depending on the relationship between the number of trajectory points T_{obs} and the number of control points K, this equation may form an overdetermined system ($T_{obs} > K$), a determined system ($T_{obs} = K$) or an underdetermined system ($T_{obs} < K$). To handle these three cases uniformly, we employ a regularized pseudo-inverse method:

$$\mathbf{C}_{init} = (\mathbf{M}^\top \mathbf{M} + \lambda \mathbf{I})^{-1} \mathbf{M}^\top \mathbf{X}_t^{tar} \tag{4}$$

Algorithm 1 BC-Attack

Input: Original trajectory $\mathbf{X}_t^{tar}$, prediction model f_θ, iterations N, learning rate η
Output: Adversarial trajectory $\tilde{\mathbf{X}}_t^{tar}$

Construct Bernstein basis matrix $\mathbf{M}$;
Compute $\mathbf{C}_{init} = (\mathbf{M}^\top \mathbf{M} + \lambda \mathbf{I})^{-1} \mathbf{M}^\top \mathbf{X}_t^{tar}$;
$\mathbf{C}_{start} = \mathbf{C}_{init} + \boldsymbol{\epsilon}_0$, where $\boldsymbol{\epsilon}_0 \sim \mathcal{U}(-\sigma_0, \sigma_0)$;
Initialize $\Delta\mathbf{C} \leftarrow \mathbf{0}$
for $i = 1$ to N **do**
  $\mathbf{C}_{adv} \leftarrow \mathbf{C}_{start} + \Delta\mathbf{C}$;
  Compute $\tilde{\mathbf{X}}_t^{tar} = \mathbf{M}\mathbf{C}_{adv}$;
  Compute loss $\mathcal{L}_{total} = \mathcal{L}_{attack} + \mathcal{L}_{phy}$;
  Update $\Delta\mathbf{C} \leftarrow \Delta\mathbf{C} - \eta \nabla_{\Delta\mathbf{C}} \mathcal{L}_{total}$;
  Clip $\Delta\mathbf{C}$ into $[-\epsilon_c, \epsilon_c]$;
end for
return $\tilde{\mathbf{X}}_t^{tar} = \mathbf{M} \cdot (\mathbf{C}_{start} + \Delta\mathbf{C})$

where λ is a small regularization coefficient (set to 10^{-6} in our experiments) to ensure numerical stability and matrix invertibility. When $T_{obs} > K$, this method is equivalent to the least squares solution, finding the control points that minimize the reconstruction error $\|\mathbf{MC} - \mathbf{X}_t^{tar}\|^2$; when $T_{obs} \leq K$, this method yields an exact or approximate solution with a smaller norm. From a geometric perspective, this step projects the original trajectory onto the smooth Bézier space, where $\mathbf{C}_{init}$ represents the optimal approximation of the original trajectory in the Bézier space.

However, using $\mathbf{C}_{init}$ directly as the optimization starting point may cause the algorithm to converge to a local optimum near the original trajectory, limiting the attack effectiveness. To enhance the exploration capability of the optimization process, we add small random perturbations to the initial control points:

$$\mathbf{C}_{start} = \mathbf{C}_{init} + \boldsymbol{\epsilon}_0, \quad \boldsymbol{\epsilon}_0 \sim \mathcal{U}(-\sigma_0, \sigma_0) \tag{5}$$

where $\boldsymbol{\epsilon}_0$ is random noise following a uniform distribution, and σ_0 is a constant controlling the perturbation magnitude. The selection of this value requires a trade-off between two aspects: perturbations that are too small may fail to effectively escape local optima, while perturbations that are too large may cause the initial trajectory to deviate too far from the original, affecting the optimization convergence speed. In our experiments, we set σ_0 to 0.1m.

C. Adversarial Optimization

We define a learnable perturbation variable $\Delta\mathbf{C} \in \mathbb{R}^{K\times 2}$, initialized to zero. At each iteration, the adversarial control points and the corresponding trajectory are computed as:

$$\mathbf{C}_{adv} = \mathbf{C}_{start} + \Delta\mathbf{C} \tag{6}$$

$$\tilde{\mathbf{X}}_t^{tar} = \mathbf{M}\mathbf{C}_{adv} \tag{7}$$

Attack Objective. The goal is to maximize the prediction error of the victim model f_θ. The attack loss is defined as:

$$\mathcal{L}_{attack} = -\text{ADE}(f_\theta(\tilde{\mathbf{X}}_t^{tar}, \mathbf{I}), \mathbf{F}_t^{tar}) = -\frac{1}{T_{pred}} \sum_{i=1}^{T_{pred}} \|\hat{\mathbf{s}}_{t+i} - f_{t+i}^{tar}\|_2 \tag{8}$$

Minimizing $\mathcal{L}_{attack}$ is equivalent to maximizing the Average Displacement Error (ADE).

Constraint Strategy. To ensure that the adversarial trajectory is both covert and physically feasible, we employ a two-stage constraint strategy to achieve this:

1) **Control Point Bound (Hard Constraint):** Since the mapping $\tilde{\mathbf{X}}_t^{tar} = \mathbf{M}\mathbf{C}_{adv}$ is linear, trajectory deviation can be implicitly constrained by bounding $\Delta\mathbf{C}$, thereby avoiding adversarial trajectories with excessive perturbations:

$$\Delta\mathbf{C} \leftarrow \text{clip}(\Delta\mathbf{C}, -\epsilon_c, \epsilon_c) \tag{9}$$

where ϵ_c is the maximum allowed perturbation for each control point coordinate (e.g., 0.5m). This clipping operation is applied after each gradient update.

2) **Kinematic Penalty (Soft Constraint):**While the control point bound ensures spatial proximity to the original trajectory, we also need to ensure that the generated trajectory obeys the physical laws governing vehicle motion. Specifically, the velocity and acceleration derived from the trajectory must remain within realistic bounds. A naive approach would be to clip the velocity and acceleration values directly in the trajectory space. However, this would break the smooth structure of the Bézier Curve, reintroducing the very problem we aimed to solve. Instead, we incorporate the kinematic constraints as a soft penalty term in the loss function:

$$\begin{aligned}\mathcal{L}_{phy} = \lambda_v \sum_{t=2}^{T_{\text{obs}}} \max(0, \|v_t\| - v_{\max})^2 \\ + \lambda_a \sum_{t=3}^{T_{\text{obs}}} \max(0, \|a_t\| - a_{\max})^2\end{aligned} \tag{10}$$

where v_t and a_t are the velocity and acceleration at frame t of the adversarial trajectory, computed from the adversarial trajectory using finite differences:

$$v_t = \frac{x_t - x_{t-1}}{\Delta t} \tag{11}$$

$$a_t = \frac{v_t - v_{t-1}}{\Delta t} \tag{12}$$

The coefficients λ_v and λ_a are positive weighting factors that control the trade-off between attack effectiveness and physical plausibility. Larger values of λ_v and λ_a impose stricter physical constraints but may reduce attack strength, while smaller values allow more aggressive attacks at the cost of potential physical implausibility. In our experiments, we set $\lambda_v = \lambda_a = 1.0$ as a balanced default. This soft constraint mechanism allows the optimizer to naturally find solutions that are both effective and physically feasible, without the discontinuities introduced by hard clipping.

Total Loss. The total optimization objective is:

$$\mathcal{L}_{total} = \mathcal{L}_{attack} + \mathcal{L}_{phy} \tag{13}$$

Guided by $\mathcal{L}_{total}$, PGD iteratively updates the control point locations $\mathbf{C}$ and ultimately converges on the optimal control point locations $\mathbf{C}_{adv}$.

TABLE I: Summary of datasets.

Name	Scenario	Map	T_{obs}	T_{pred}	Freq. (Hz)
Apolloscape [15]	Urban	×	6	6	2
nuScenes [16]	Urban	✓	4	12	2

V. EXPERIMENTS

A. Experimental Setting

Datasets. We evaluate BC-Attack on two public datasets: Apolloscape [15], and nuScenes [16]. The dataset characteristics are summarized in Table I. We select the history length (T_{obs}) and prediction length (T_{pred}) following the official

TABLE II: Average prediction error under normal and adversarial attacks

Model	Dataset	Method	ADE	FDE	LDR
GRIP	nuScenes	Normal	4.35	10.40	2.4%
		SA-Attack	6.03	13.55	2.5%
		Opt-Attack	6.98	15.41	2.6%
		BC-Attack	**7.91**	**16.93**	**3.7%**
	Apolloscape	Normal	1.66	3.18	0.7%
		SA-Attack	3.11	5.82	4.8%
		Opt-Attack	**6.41**	**10.88**	4.3%
		BC-Attack	5.98	10.22	**5.5%**
FQA	nuScenes	Normal	4.20	10.04	0.0%
		SA-Attack	5.66	12.53	10.6%
		Opt-Attack	6.52	14.17	4.1%
		BC-Attack	**7.48**	**15.75**	**17.5%**
	Apolloscape	Normal	2.04	3.83	0.2%
		SA-Attack	2.81	5.14	39.7%
		Opt-Attack	**6.73**	**11.71**	20.5%
		BC-Attack	6.63	11.08	**45.7%**
Trajectron++	nuScenes	Normal	3.90	11.05	1.9%
		SA-Attack	8.04	19.27	45.0%
		Opt-Attack	8.57	19.94	48.7%
		BC-Attack	**9.19**	**21.13**	**49.8%**
	Apolloscape	Normal	1.28	2.94	0.0%
		SA-Attack	3.80	7.19	35.6%
		Opt-Attack	7.37	**13.71**	22.3%
		BC-Attack	**7.80**	13.19	**42.5%**
Trajectron++(map)	nuScenes	Normal	3.12	6.20	0.9%
		SA-Attack	6.72	16.48	26.8%
		Opt-Attack	8.07	18.63	28.0%
		BC-Attack	**8.66**	**19.71**	**30.6%**

recommendations. We randomly select 100 scenarios as test cases from each dataset.

Models. We evaluated three state-of-the-art trajectory prediction models: GRIP [5], FQA [6], Trajectron++ [4]. Note that Trajectron++ generates multimodal trajectory predictions with probabilities, and we selected the highest-probability trajectory as the final prediction. Additionally, Trajectron++ can incorporate semantic map information as input, which is only available in nuScenes. Therefore, we evaluated Trajectron++ (w/o map) on all datasets, while Trajectron++ (map) was evaluated exclusively on nuScenes. For all models, we use officially released pretrained weights or train with recommended hyperparameters.

Attack methods. We select two representative attacks as baselines: the trajectory reconstruction-based attack proposed by Yin et al. [9] and the perturbation optimization-based attack proposed by Zhang et al. [7], henceforth referred to as *SA-Attack* and *Opt-Attack*, respectively.

Metrics. We employ two widely used metrics, ADE and FDE [32], to measure the impact of adversarial attacks on prediction accuracy, and propose a new metric, LDR, to evaluate the degree of lane deviation in predicted trajectories:

Average Displacement Error (ADE): Mean ℓ_2 distance between the ground truth and predicted trajectories.

Final Displacement Error (FDE): ℓ_2 distance between the predicted final position and the ground truth final position at the prediction horizon T_{pred}.

Lateral Deviation Ratio (LDR): The proportion of predicted time steps whose lateral deviation from the reference centerline exceeds half of the lane width (1.85m).

Implementation details. For SA-Attack, Opt-Attack and our BC-Attack, we adopt PGD-based white-box optimization. For each scenario, we conduct five independent repeated experiments and report the result with the best attack performance. The learning rate is set to 0.1 and the maximum number of iterations is 200. The perturbation bound on control points is set to 0.5m.

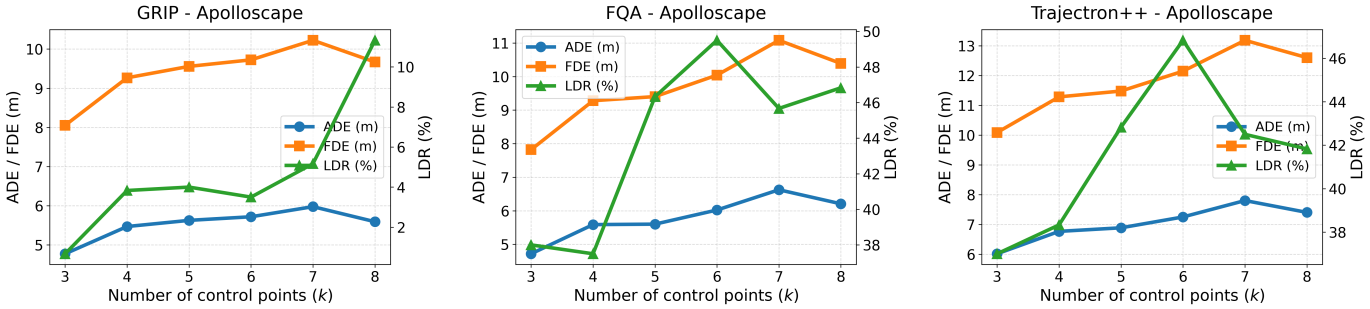
B. Attack Results

Trajectory prediction under attacks. First, we analyzed the effectiveness of three attack methods for different combinations of models and datasets, and Table II shows the average prediction error before and after the attack (normal, unperturbed). BC-Attack improved the average ADE/FDE of trajectory prediction by 210%/158% compared to normal prediction, and achieved the highest or second-highest ADE/FDE across different model and dataset combinations, indicating that BC-Attack's attack effect is better than the baseline. In addition, we used LDR to measure the impact of the attack method on future predictions. BC-Attack had the highest LDR in all combinations, meaning that more frames in future predictions had lateral deviations exceeding the threshold (half the lane width, 1.85m), which had a greater impact on vehicles on both sides, potentially causing the victim vehicle to brake suddenly or swerve, thus leading to traffic accidents. The experimental results also show that Trajectron-map outperformed Trajectron before and after the attack, indicating that semantic maps do indeed significantly improve the accuracy and robustness of trajectory prediction. Additionally, it can be observed that the attack metrics for all attacks are better on the nuScenes dataset than on the Apolloscape dataset. This is because the predicted trajectory length of the nuScenes dataset is twice that of the Apolloscape dataset, and the bias of later time frames is greater than that of earlier frames, thus raising the average value and resulting in a higher ADE/FDE ratio.

Number of control points. We conducted a systematic comparative experiment on the number of control points $k \in \{3, 4, 5, 6, 7, 8\}$ in the BC-Attack, and the results are shown in Fig. 2 and Fig. 3.

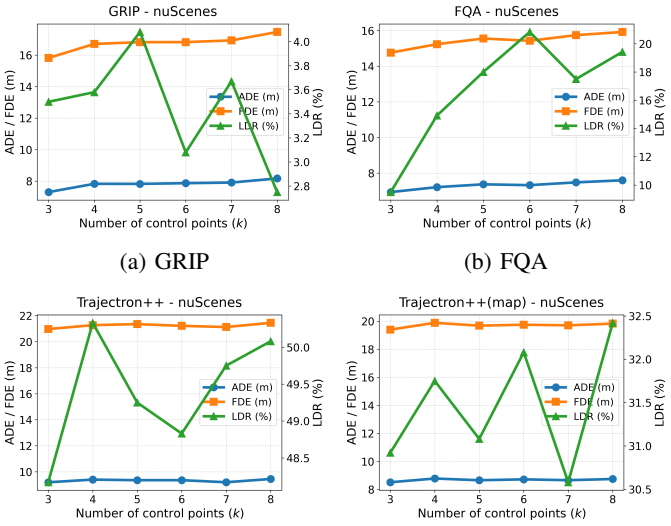
Overall, as the number of control points k increases, both ADE and FDE show a slight upward trend. When $k = 8$, compared to $k = 3$, the average ADE improves by approximately 11.0% and the average FDE improves by approximately 14.4%. This phenomenon is consistent with the basic properties of Bézier Curves: an increase in the number of control points means an increase in the curve order, which theoretically provides richer shape representation and a larger optimization space. However, due to the relatively short length of the historical trajectories in this experiment (the observation lengths for nuScenes and Apolloscape are 4 and 6 time steps, respectively, see Table I)), the actual optimizable "operational space" of the curve is limited, therefore the error increase is not significant.

Notably, the growth magnitude of ADE/FDE for BC-Attack on the Apolloscape dataset (See Fig. 2) is significantly larger



(a) GRIP (b) FQA (c) Trajectron++

Fig. 2: Impact of the number of control points (k) on BC-Attack performance on the Apolloscape dataset. Left axis: ADE/FDE (m); Right axis: LDR (%).



(a) GRIP (b) FQA

(c) Trajectron++ (d) Trajectron++ (Map)

Fig. 3: Impact of the number of control points (k) on BC-Attack performance on the nuScenes dataset. Left axis: ADE/FDE (m); Right axis: LDR (%).

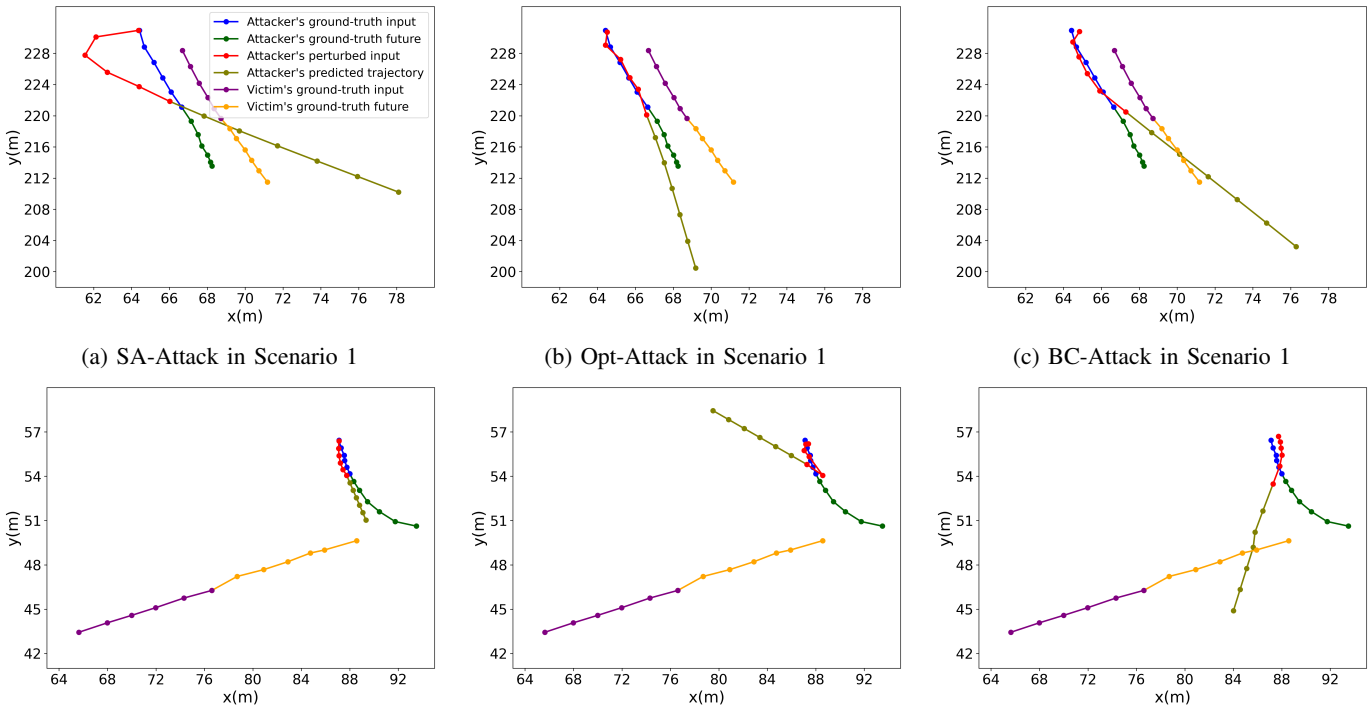
than on the nuScenes dataset ((See Fig. 3)). Specifically, on Apolloscape, when k increases from 3 to 7, GRIP's ADE rises from 4.77 to 5.98 (25.4% increase) and FDE rises from 8.05 to 10.22 (27.0% increase); whereas on nuScenes, GRIP's ADE rises from 7.30 to 7.91 (8.4% increase) and FDE rises from 15.82 to 16.93 (7.0% increase). This significant discrepancy can be explained through the matching relationship between trajectory length and control point quantity: nuScenes has a historical trajectory length of 4 timesteps, which is relatively short, limiting the manipulability of control points on adversarial trajectories. Since each control point exerts a global influence on the entire trajectory through Bernstein basis functions, when trajectory points are few, the shape variation space that control points can "pull" is quite limited. Experimental data show that on nuScenes, attack effectiveness approaches saturation at $k = 3$ or $k = 4$ (e.g., GRIP has an ADE of 7.83 at $k = 4$, increasing only to 8.17 at $k = 8$). Further increasing the number of control points has minimal impact on trajectory shape modification while increasing training time.

In contrast, Apolloscape has a historical trajectory length of 6 timesteps, providing more manipulable trajectory points. In this scenario, increasing the number of control points can more effectively leverage the shape expressiveness of Bézier Curves. Experimental results show that most models on Apolloscape achieve optimal attack performance at $k = 7$, where the number of control points approximates the trajectory length, allowing Bézier Curves to sufficiently fit local trajectory variations while avoiding over-parameterization. When k continues to increase to 8, the improvement in attack performance is not significant (GRIP's ADE even slightly decreases to 5.59), but the training time further increases.

This observation reveals an important design principle: when k is approximately equal to or slightly greater than the trajectory length T_{obs}, the Bézier Curve has sufficient degrees of freedom to capture trajectory details without introducing too many redundant parameters that would reduce optimization efficiency.

Regarding LDR, as the number of control points increases, the LDR of each model on different datasets shows a slight fluctuation or a slow increase. For example, in the FQA-Apolloscape combination, the LDR fluctuates from 38.0% when $k = 3$ to 45.7% when $k = 7$; in the Trajectron++-nuScenes combination, the LDR slightly increases from 48.2% to 50.1%. Although the changes in LDR are not as pronounced as those in ADE/FDE, BC-Attack significantly outperforms the baseline attack method in all configurations. This suggests that appropriately increasing the number of control points helps to amplify the lateral offset in future time steps (a deviation from the lane centerline exceeding half the lane width of 1.85m) while maintaining trajectory smoothness, thereby enhancing the interference effect on the planning and avoidance modules of the autonomous driving system.

Case study. In Fig. 4, we visualize the adversarial trajectories generated by SA-Attack [9], Opt-Attack [7], and BC-Attack in two scenarios. We demonstrate that our method can generate both realistic and effective adversarial trajectories.



(a) SA-Attack in Scenario 1 (b) Opt-Attack in Scenario 1 (c) BC-Attack in Scenario 1

(d) SA-Attack in Scenario 2 (e) Opt-Attack in Scenario 2 (f) BC-Attack in Scenario 2

Fig. 4: Visual comparison of adversarial trajectories generated by three attack methods. (a)-(c) represent the trajectories in Scenario 1, while (d)-(f) represent Scenario 2. The legend in (a) applies to all subfigures.

In scenario 1, our method generates a smooth adversarial trajectory with small perturbations, causing the predicted trajectory to deviate significantly from the real trajectory. In the first frame after the attack, the distance between the victim vehicle and the attacker-controlled vehicle reaches a minimum of 0.72m. At this distance, the victim vehicle is highly likely to trigger emergency braking due to the close proximity, leading to a rear-end collision. SA-Attack reconstructs the trajectory using a continuous curvature model, which also generates an effective attack, but the perturbation is too obvious. Opt-Attack causes relatively small lateral errors and does not threaten the safety of the victim vehicle, but it successfully misleads the prediction of the attacker vehicle's future intentions, mistaking the realistic gradual deceleration for maintaining speed. In scenario 2, our method also adapted well to low-speed conditions, but the predicted trajectory after the attack showed a significant deviation, predicting a collision with the victim vehicle between the third and fourth frames. In contrast, SA-Attack did generate a smooth trajectory, but the offset was weak, limiting the attack effect. Opt-Attack generated a jagged adversarial trajectory that was physically impossible to implement, making it too anomalous and easily detected by defensive methods.

VI. CONCLUSIONS

This paper proposes a novel adversarial attack method named BC-Attack (Bézier Curves Attack) to investigate the security of autonomous driving trajectory prediction models. We innovatively employ Bézier Curves for parametric modeling of trajectories, shifting the optimization objective from discrete trajectory points to the control points of the curve. This approach ensures continuous curvature and smoothness in the generated adversarial trajectories while maintaining attack effectiveness. Extensive comparative experiments conducted on nuScenes and Apolloscape datasets across multiple mainstream trajectory prediction models demonstrate that BC-Attack increases the average prediction error by 210% compared to normal prediction. Visualization results further validate our method's capability to generate natural, smooth, and deceptive adversarial trajectories. This study reveals the vulnerability of current trajectory prediction models to adversarial attacks, providing valuable research insights for enhancing the safety and robustness of autonomous driving systems.

REFERENCES

[1] Autoware.ai. https://www.autoware.ai/, 2020.
[2] Baidu apollo. http://apollo.auto/, 2021.
[3] Y. Yuan, X. Weng, Y. Ou, and K. Kitani, "AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting," in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, Oct. 2021, pp. 9793–9803.
[4] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in Computer Vision–ECCV 2020, 16th European Conf., Glasgow, UK, 2020, pp. 683–700.
[5] X. Li, X. Ying, and M. C. Chuah, "GRIP: Graph-based Interaction-aware Trajectory Prediction," in 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE, Oct. 2019.

[6] N. Kamra, H. Zhu, D. K. Trivedi, M. Zhang, and Y. Liu, "Multi-agent trajectory prediction with fuzzy query attention," in Advances in Neural Information Processing Systems, vol. 33, pp. 22530–22541, 2020.

[7] Q. Zhang, S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao, "On Adversarial Robustness of Trajectory Prediction for Autonomous Vehicles," in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2022, pp. 15138–15147.

[8] K. Tan, J. Wang, and Y. Kantaros, "Targeted adversarial attacks against neural network trajectory predictors," in Learning for Dynamics and Control Conference, 2023, pp. 431–444.

[9] H. Yin, J. Li, P. Zhen, and J. Yan, "SA-Attack: Speed-adaptive stealthy adversarial attack on trajectory prediction," in 2024 IEEE Intelligent Vehicles Symposium (IV), IEEE, Jun. 2024, pp. 1772–1778.

[10] X. Bai, C. Tian, W. Xia, Z. Ma, H. Yu, and Y. Ren, "EL-Attack: Explicit and Latent Space Hybrid Optimization based General and Effective Attack for Autonomous Driving Trajectory Prediction," in 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, Jun. 2025, pp. 3582–3590.

[11] Y. Cao, C. Xiao, A. Anandkumar, D. Xu, and M. Pavone, "AdvDO: Realistic Adversarial Attacks for Trajectory Prediction," in Lecture Notes in Computer Science, Springer Nature Switzerland, 2022, pp. 36–52.

[12] J. Fan, Z. Wang, and G. Li, "Adversarial Attack on Trajectory Prediction for Autonomous Vehicles with Generative Adversarial Networks," in 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Oct. 2024, pp. 1026–1031.

[13] Y. Dong et al., "Safe Driving Adversarial Trajectory Can Mislead: Toward More Stealthy Adversarial Attack Against Autonomous Driving Prediction Module," ACM Transactions on Privacy and Security, vol. 28, no. 2, pp. 1–28, Feb. 2025.

[14] Schumann, Julian F., et al. "Realistic Adversarial Attacks for Robustness Evaluation of Trajectory Prediction Models via Future State Perturbation." arXiv preprint arXiv:2505.06134, 2025.

[15] X. Huang et al., "The ApolloScape Dataset for Autonomous Driving," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, Jun. 2018.

[16] H. Caesar et al., "nuScenes: A Multimodal Dataset for Autonomous Driving," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2020.

[17] Y. Lou, Y. Zhu, Q. Song, R. Tan, C. Qiao, W. B. Lee, and J. Wang, "A First Physical-World Trajectory Prediction Attack via LiDAR-induced Deceptions in Autonomous Driving," in 33rd USENIX Security Symposium (USENIX Security 24), 2024, pp. 6291–6308.

[18] Y. Yu, W. Han, L. Wu, B. Liu, E. Wang, and Z. Zhang, "Enduring, Efficient and Robust Trajectory Prediction Attack in Autonomous Driving via Optimization-Driven Multi-Frame Perturbation Framework," in 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2025, pp. 17229–17238.

[19] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, "Learning to Predict Vehicle Trajectories with Model-based Planning," in Proceedings of the 5th Conference on Robot Learning, 2022, pp. 1035–1045.

[20] H. Song, W. Ding, Y. Chen, S. Shen, M. Y. Wang, and Q. Chen, "PiP: Planning-Informed Trajectory Prediction for Autonomous Driving," in Lecture Notes in Computer Science, Springer International Publishing, 2020, pp. 598–614.

[21] H. Cui et al., "Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks," in 2019 International Conference on Robotics and Automation (ICRA), IEEE, May 2019, pp. 2090–2096.

[22] Z. Huang, H. Liu, J. Wu, and C. Lv, "Differentiable Integrated Motion Prediction and Planning With Learnable Cost Function for Autonomous Driving," IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 11, pp. 15222–15236, Nov. 2024.

[23] J. Gu et al., "ViP3D: End-to-End Visual Trajectory Prediction via 3D Agent Queries," in 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2023.

[24] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, "TraPHic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2019, pp. 8475–8484.

[25] C. Choi, J. H. Choi, J. Li, and S. Malla, "Shared Cross-Modal Trajectory Prediction for Autonomous Driving," in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2021, pp. 244–253.

[26] N. Shafiee, T. Padir, and E. Elhamifar, "Introvert: Human Trajectory Prediction via Conditional 3D Attention," in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2021, pp. 16815–16825.

[27] L. Shi, L. Wang, C. Long, S. Zhou, M. Zhou, Z. Niu, and G. Hua,"Sgcn: Sparse graph convolution network for pedestrian trajectory prediction," in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2021, pp. 8994–9003.

[28] J. Wang, P. Wang, C. Zhang, K. Su, and J. Li, "F-Net: Fusion Neural Network for Vehicle Trajectory Prediction in Autonomous Driving," in ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Jun. 2021, pp. 4095–4099.

[29] Y. Lin et al., "JarvisIR: Elevating Autonomous Driving Perception with Intelligent Image Restoration," in 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2025, pp. 22369–22380.

[30] X. Guo, F. Jiang, Q. Chen, Y. Wang, K. Sha, and J. Chen, "Deep learning-enhanced environment perception for autonomous driving: MD-Net with CSP-DarkNet53," Pattern Recognition, vol. 160, p. 111174, Apr. 2025.

[31] D. Kang, J. Lee, and H. Baek, "Real-time scheduling for multi-object tracking tasks in regions with different criticalities," Journal of Systems Architecture, vol. 160, p. 103349, Mar. 2025.

[32] R. Chandra, U. Bhattacharya, C. Roncal, A. Bera, and D. Manocha, "RobustTP: End-to-End Trajectory Prediction for Heterogeneous Road-Agents in Dense Traffic with Noisy Sensor Inputs," in ACM Computer Science in Cars Symposium, ACM, Oct. 2019, pp. 1–9.

[33] P. Pandunata and S. M. H. Shamsuddin, "Differential Evolution Optimization for Bezier Curve Fitting," in 2010 Seventh International Conference on Computer Graphics, Imaging and Visualization, IEEE, Aug. 2010, pp. 68–72.

[34] A. Tharwat, M. Elhoseny, A. E. Hassanien, T. Gabel, and A. Kumar, "Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm," Cluster Computing, vol. 22, no. S2, pp. 4745–4766, Mar. 2018.

[35] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "ABCNet: Real-Time Scene Text Spotting With Adaptive Bezier-Curve Network," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2020, pp. 9806–9815.