Kinematics-aware Trajectory Generation and Prediction with Latent Stochastic Differential Modeling

Ruochen Jiao*[1], Yixuan Wang*[1], Xiangguo Liu[1], Simon Sinong Zhan[1], Chao Huang[2], Qi Zhu[1]

***Abstract*— Trajectory generation and trajectory prediction are two critical tasks in autonomous driving, which generate various trajectories for testing during development and predict the trajectories of surrounding vehicles during operation, respectively. In recent years, emerging data-driven deep learning-based methods have shown great promise for these two tasks in learning various traffic scenarios and improving average performance without assuming physical models. However, it remains a challenging problem for these methods to ensure that the generated/predicted trajectories are physically realistic. This challenge arises because learning-based approaches often function as opaque black boxes and do not adhere to physical laws. Conversely, existing model-based methods provide physically feasible results but are constrained by predefined model structures, limiting their capabilities to address complex scenarios. To address the limitations of these two types of approaches, we propose a new method that integrates kinematic knowledge into neural stochastic differential equations (SDE) and designs a variational autoencoder based on this latent kinematics-aware SDE (*LK-SDE*) to generate vehicle motions. Experimental results demonstrate that our method significantly outperforms both model-based and learning-based baselines in producing physically realistic and precisely controllable vehicle trajectories. Additionally, it performs well in predicting unobservable physical variables in the latent space.**

I. Introduction

Trajectory prediction and generation are two critical tasks for autonomous vehicles. First, as a key component in the autonomous driving pipeline, the *trajectory prediction* module predicts the future trajectories of surrounding vehicles based on their recent trajectory histories (as observed by the ego vehicle) and the map information. The prediction result provides a safe operation space for downstream behavioral-level decision-making and motion planning [1]–[4] and is critical for vehicle safety during operation. Then, given the long-tailed nature of real traffic scenarios, the important *trajectory generation* task generates additional synthetic but realistic trajectories to augment the trajectory dataset collected in operation, for testing and optimizing the downstream planning module [5], [6]. For instance, as shown in Fig. 1, we can convert a common and simple scenario to various challenging scenarios in the simulation by generating diverse trajectories and using them to test the reaction of the autonomous vehicle.

*Contribute equally to this work.

[1]Ruochen Jiao, Yixuan Wang, Xiangguo Liu, Simon Sinong Zhan, and Qi Zhu are with the Department of Electrical and Computer Engineering, Northwestern University, IL, USA. {ruochen.jiao,yixuanwang2024}@u.northwestern.edu

[2]Chao Huang is with the School of Electronics and Computer Science, the University of Southampton, UK.

In the literature, most works focus on improving the average accuracy for trajectory prediction and enumerating various scenarios for trajectory generation. However, the predicted/generated trajectories may not be realistic or even physically feasible in real traffic scenarios, and could lead to inferior training of the planning module and reduced capability of addressing safety-critical scenarios in practice. It is thus very important to ensure that the predicted/generated trajectories not only reflect the rich contextual factors, including other vehicles and HD maps but also *conform to traffic rules and fundamental laws of physics*.

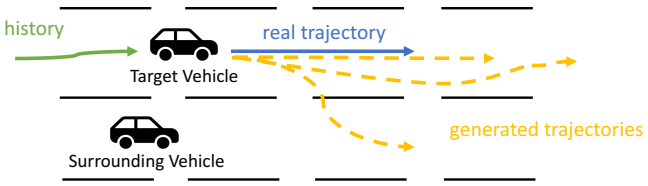

Fig. 1: Generating diverse, physically realistic, and controllable trajectories is important for critical scenario augmentation and vehicle motion prediction.

It is, however, challenging for the trajectory predictor and generator to learn the representation of high-dimensional context while aligning with physical constraints. Traditional model-based techniques leverage simplified kinematic models, e.g., bicycle models, for trajectory generation. However, these methods often provide a very rough estimation of the vehicle's real motion and oversimplify the environment model without considering the surroundings, leading to coarse-grained and maladaptive generated trajectories in complex traffic scenarios. On the other hand, learning-based generative models may effectively perceive the high-dimensional environment by learning from trajectory data and map context, as shown in recent advancements in trajectory prediction [7]–[9] and generation [10]–[13]. However, due to the lack of consideration of physical models, these generative techniques have little control over finer-grained vehicle-level kinematics and may produce physically unrealistic trajectories in real traffic scenarios.

In this paper, we aim to address the fundamental challenge in **simultaneously considering the complex surrounding environment and the vehicle physical model** for trajectory generation and prediction. The major difficulty is from the conflict between high-dimensional environment (captured by deep learning) and low-dimensional physics. To address this, we utilize **latent kinematics-aware neural stochas-**

tic differential equations *(LK-SDE)* to bridge the model-based kinematics and high-dimensional representations for road contexts in the latent space of a variational autoencoder (VAE) [14]. Specifically, we first extract the graph convolutional network (GCN) representation from the HD maps and environment, which convey the information of the environment contexts. Together with a physical model, the GCN representation is then used to train a neural SDE. Such kinematics-aware SDE is optimized and calculated in the latent space of the VAE, providing additional critical information for prediction and precise control for decoding the trajectory generation. It is worth noting that this architecture can be added to various scenario augmentation and trajectory prediction frameworks to enhance physical realism and controllability.

Our contributions in this work are summarized as follows:

- We design a trajectory generator based on kinematics-guided SDE in the latent space, which effectively *embeds physical constraints into deep learning models*.
- For the trajectory generation task, compared with pure deep learning-based and model-based methods, our method can *generate more physically realistic and controllable augmented trajectories* by manipulating the kinematics-aware latent space.
- For the trajectory prediction task, our method can jointly *predict realistic trajectories and important kinematic states* that are difficult to observe directly.
- Extensive experiments show that our method outperforms baseline methods across various metrics. These improvements will benefit the augmentation of safety-critical scenarios and the prediction of future motions.

This paper is organized as follows. Section II introduces the background of trajectory generation and prediction and neural SDE. Section III presents the design of our *LK-SDE* based VAE for trajectory generation and prediction. Experiment results are shown in Section IV and Section V concludes the paper.

II. Background

A. Trajectory Generation and Prediction

Trajectory generation or augmentation plays an important role in evaluating and optimizing the decision-making module in autonomous driving. Many works use various deep learning methods to generate trajectories and scenarios. For instance, the work in [11] proposes a VAE-conditioned method to bridge safe- and collision-driving data to generate the whole risky scenario, but it cannot control agent-level trajectories. The work in [10] designs a GAN-based method – RouteGAN to generate diverse trajectories for every single agent, and the trajectory is controlled by a style variable. Some recent approaches [12], [13], [15] further utilize domain knowledge such as causal relations and traffic priors to generate useful traffic scenarios. However, these methods mainly focus on scenario-level generation. For trajectory generation of a single vehicle, these works still rely on pure deep learning methods or model-based methods. The latent spaces of these generative methods are not well modeled or explained for a single vehicle, especially at the kinematics or dynamics level. The models only have coarse and limited control over the generated trajectories, which often lead to physically unrealistic and uncontrollable trajectories.

Recent works applied advanced deep learning techniques to learn the representations of agents' trajectories and road contexts. Graph neural networks [7], [16], transformer [9], [17], and diffusion models [18] are used to extract context features. The approach in [19] adds a bicycle model after the neural feature extractor to decode the trajectories but their pure model-based decoder still suffers from the oversimplification of the vehicle motion. In this work, we aim to combine the powerful representations from deep learning models with kinematics knowledge in the latent space.

B. Neural Stochastic Differential Equation

The physical dynamics of many real-world systems can be modeled as a discrete-time SDE [20] with the consideration of uncertainty and stochasticity.

Definition 1: **(Stochastic Differential Equation (SDE))**: An SDE is a differential equation that contains stochastic processes, which can be expressed as

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t) + g(\mathbf{s}_t)\Delta W_t, \tag{1}$$

where $\mathbf{s}_t \in \mathbb{R}^n$ is the system state, $f : \mathbb{R}^n \to \mathbb{R}^n$ denotes a drift function, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times d}$ represents a diffusion function. $\Delta W_t = W(t+1) - W(t)$, where $W(t) \in \mathbb{R}^d$ is the Brownian Motion (also known as Wiener Process) [21] for encoding the stochasticity in the systems. The Brownian Motion has the following properties:

- $W(0) = 0$,
- $W(t)$ is almost surely continuous,
- $W(t_1) - W(t_0) \sim \mathcal{N}(0, t_1 - t_0)$, where $\mathcal{N}(0, t_1 - t_0)$ is the Gaussian distribution with 0 mean and $t_1 - t_0$ variance.

Definition 2: **(Neural SDE)**: A neural SDE is an SDE with its drift function $f(\mathbf{s}_t)$ and diffusion function $g(\mathbf{s}_t)$ expressed and parameterized by deep neural networks, e.g., $f_{\theta_0}(\mathbf{s}_t), g_{\theta_1}(\mathbf{s}_t)$ [22]:

$$\hat{\mathbf{s}}_{t+1} = f_{\theta_0}(\hat{\mathbf{s}}_t) + g_{\theta_1}(\hat{\mathbf{s}}_t)\Delta W_t. \tag{2}$$

To learn such neural network representations, a typical way is to sample the real physical environment as Eq. (1) and neural SDE as Eq. (2) to generate the trajectory data $\tau = \{\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_T\}$ and $\hat{\tau} = \{\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1, \cdots, \hat{\mathbf{s}}_T\}$, respectively. Then fit the real-world trajectory τ to the neural networks by reducing the following loss function:

$$\min_{\theta_0,\theta_1} \mathcal{L}(\tau, \hat{\tau}) = \min_{\theta_0,\theta_1} -\sum_{t=0}^{T} \log\left(P\left(\mathbf{s}_t \mid \mathcal{N}(\hat{\mathbf{s}}_t, g_{\theta_1}(\hat{\mathbf{s}}_t))\right)\right),$$

where $\mathcal{L}$ is the maximum likelihood loss, T is the time length, and $P\left(\mathbf{s}_t \mid \mathcal{N}(\hat{\mathbf{s}}_t, g_{\theta_1}(\hat{\mathbf{s}}_t))\right)$ is the likelihood probability of the observed $\mathbf{s}_t$ under the normal distribution $\mathcal{N}(\hat{\mathbf{s}}_t, g_{\theta_1}(\hat{\mathbf{s}}_t))$ of the neural SDE at time t.

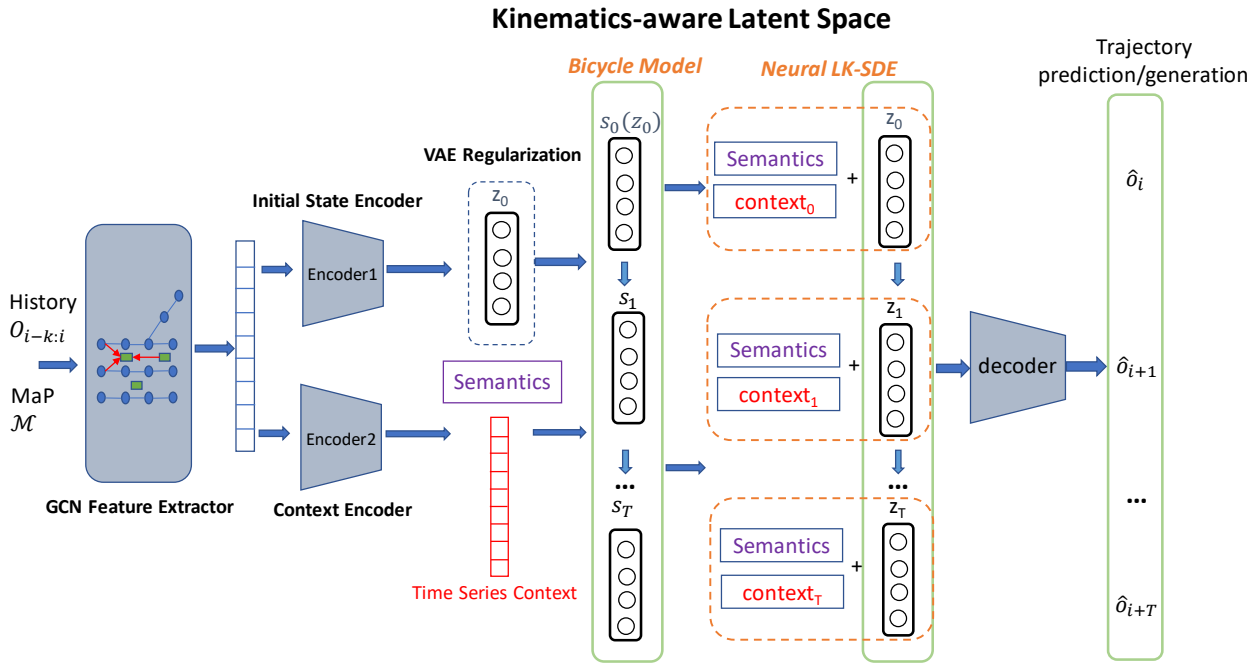

Fig. 2: The overall design of our kinematics-aware trajectory generator. The GCN and two individual encoders consume the driving historical trajectories $o_{i-k:i}$ and map information $\mathcal{M}$. They extract and generate the latent initial state z_0, global semantic feature $\mathbf{sem}$, and context feature per step $\mathbf{ctx}_t$ for the latent space. Within the latent space, we learn a kinematics-aware neural SDE guided by a physical bicycle model and then decode the latent vectors $\mathbf{z}_i(i = 0, \cdots, T)$ to the output vehicle motion trajectory $\hat{o}_{i:i+T}$. Our neural *LK-SDE* is guided by the kinematic bicycle model during training to learn physical knowledge for physically feasible and controllable trajectory generation and prediction.

III. Our LK-SDE Methods

A. Overall Design

Problem Set Up: We study the trajectory prediction and generation tasks that aim to predict and generate future two-dimensional trajectory waypoints given the driving histories of the driving cars as well as the map information. We are given a dataset of trajectory observations and map information as $(o_i, \mathcal{M})(i = 0, \cdots, N)$. Assuming at timestamp i, the input for our entire model is a k-length two-dimensional history trajectories $o_{i-k:i}$ of vehicles (including both target and surrounding vehicles) and the $\mathcal{M}$ graph of HD maps. The output of the model is the generated or predicted future trajectories $\hat{o}_{i:i+T}$, where T is the prediction and generation horizon in the future.

Overview: Our method is a learning-based VAE approach with kinematics-aware latent space guided/constrained by the kinematics knowledge from a bicycle model. The overall architecture of the proposed method is shown in Fig. 2. We first feed the trajectories of vehicles $o_{i-k:i}$ and map contexts $\mathcal{M}$ into a GCN-based feature extractor to learn representations of HD maps and interactions between vehicles. The extracted features are further processed by two encoders – one encoder converts the representations to a four-dimensional latent initial state, and the other encoder will generate a global semantic vector and finer-grained time-series contexts for future steps in the latent space. As a model-based approach, our neural *LK-SDE* takes the semantic vector, corresponding context, and the initial state from learning as inputs and rolls out the latent states step by step via learned latent dynamics. The neural *LK-SDE* states are guided to be close to the bicycle-model latent states and thus we embed the kinematics knowledge from the bicycle model into the latent dynamics. Finally, a simple fully-connected neural network will decode the latent states into vehicle trajectory space. The bicycle model guided latent dynamics in our VAE-like approach contribute to more physically feasible and controllable trajectory learning due to the latent space constraints, compared to existing learning-based approaches.

The algorithm of our approach is shown in Algorithm 1. We introduce the details of each submodule in the following.

B. Encoders for Context Extraction and Embedding

Similar to [7], we use a one-dimensional convolutional network to model history trajectories, for its effectiveness in extracting multi-scale features and efficiency in parallel computing. Multiple graph neural networks G are utilized to learn the interactions among agents and lane nodes [7]. After fusing the features x of graphs and agents, we have two encoders (E_s, E_c) to generate contexts and the initial state $\mathbf{z}_0(\mathbf{s}_0)$ for our *LK-SDE*, as shown in Fig. 2. A ResNet [23] is applied in the context encoder to further generate global semantics $\mathbf{sem}$ and time-series local contexts $\mathbf{ctx}_{1:T}$ for

Algorithm 1: Optimization Pipelines

Initialize: feature extractor G, initial state encoder E_s, context encoder E_c, decoder D, *LK-SDE*(f_{θ_0},g_{θ_1}), and bicycle-model SDE($h(\cdot,\pi)$,g_{θ_1})
Input: past trajectories $o_{i-k:i}$ and map graph $\mathcal{M}$.
for each batch **do**
  Let features $x = G(o_{i-k:i}, \mathcal{M})$.
  Let initial kinematic vectors $\mathbf{z}_0 = \mathbf{s}_0 = E_s(x)$.
  Let global semantics and time series contexts $\mathbf{sem}$, $\mathbf{ctx}_{1,2,\ldots,T} = E_c(x)$.
  Update the G and E_s by the regularization loss L_{reg} in Eq. (3).
  for t in **range**(T) **do**
    LK-SDE computes $\mathbf{z}_{t+1} = f_{\theta_0}(\mathbf{z}_t, \mathbf{ctx}_t, \mathbf{sem}) + g_{\theta_1}(\mathbf{z}_t)\Delta W_t$.
    Bicycle model SDE computes $\mathbf{s}_{t+1} = h(\mathbf{s}_t, \pi) + g_{\theta_1}(\mathbf{s}_t)\Delta W_t$.
  end for
  Update the *LK-SDE* ($f_{\theta_0}, g_{\theta_1}$) by the kinematic loss L_{kin} in Eq. (6).
  The decoder projects latent vectors into trajectory space $\hat{o}_{i:i+T} = D(\mathbf{z}_0, \mathbf{z}_1, \cdots, \mathbf{z}_T)$.
  Update the G, E_s, E_c, *LK-SDE* (f_{θ_0},g_{θ_1}) and π by the prediction loss L_{pred} in Eq. (7).
end for

future time steps $i + (1, \cdots, T)$, which is expressed as:

$$x = G(o_{i-k:i}, \mathcal{M}), \quad \mathbf{z}_0 = E_s(x),$$
$$\mathbf{sem}, \mathbf{ctx}_{1:T} = E_c(x).$$

Therefore, for each time step in the prediction horizon $t \in [1, T]$, we will have eight-dimensional local contexts $\mathbf{ctx}_t$ and four-dimensional global semantics $\mathbf{sem}$. The latent initial state $\mathbf{z}_0$ by the initial state encoder in Fig. 2 serves as the starting point for our neural *LK-SDE* and bicycle model within the latent space. $\mathbf{z}_0$ is regularized to follow a Gaussian distribution by minimizing the Kullback–Leibler divergence as shown in Eq. (3):

$$L_{reg} = \mathcal{KL}(q(\mathbf{z}_0|x)||p(\mathbf{z}_0)), \tag{3}$$

where x is the input features, $p(\mathbf{z}_0)$ represents the targeted Gaussian distribution of the latent initial state, and q represents the posterior distribution from the initial state encoder. The initial latent states $\mathbf{z}_0$ will be the input for the following *LK-SDE*, and the global semantics $\mathbf{sem}$ as well as local context per step $\mathbf{ctx}_t$ will be the condition.

C. Latent Kinematics-Aware SDE Modeling

The generation and prediction of vehicle trajectories rely on understanding the inherent dynamics and physical laws governing these trajectories. Consequently, this compels us to focus on acquiring a latent space attuned to the kinematics. Specifically, within the scope of this work, our objective is to acquire a *LK-SDE* for motion prediction and generation. This involves not only optimizing the loss function based on the ground truth label but also incorporating supervision from an explicit bicycle model [24], to serve as a constraint and guidance for the *LK-SDE*. Specifically, the kinematics-aware latent space modeling involves two SDEs during training – one is a learnable bicycle-model-based SDE to generate the kinematics based on the most recent states, and the other is the neural *LK-SDE* that we optimized to both learn from the loss function of the output and learn the kinematics from the bicycle model SDE. The detailed kinematics-guided dual SDE learning process is illustrated in Fig. 3.

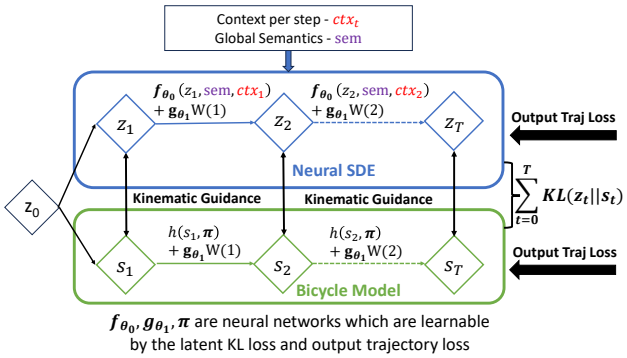

Fig. 3: During the training, in the latent space, the bicycle model SDE guides our neural *LK-SDE* to follow the kinematics by minimizing the KL divergence between the solutions of two SDEs. The kinematic loss function L_{kin} is explained in Eq. (6). f_{θ_0}, g_{θ_1} are the neural networks for *LK-SDE* that are optimized from the bicycle model in Eq. (6) and output loss function in Eq. (7), while π in the bicycle model is optimized by the output loss function in Eq. (7).

1) Learnable Bicycle Model SDE: The bicycle-model-based SDE is designed to infuse a comprehensive understanding of physics into our neural *LK-SDE*. Over time, the latent state evolves according to this SDE, enabling us to capture a series of kinematics in the latent space of the VAE. This latent trajectory is subsequently decoded to produce the final task output, thereby increasing the likelihood of adhering to the specified physical constraints.

We assume that the bicycle model SDE has $h(\mathbf{s}_t, \pi)$ as its drift function, where we differentiate by using $\mathbf{s}_t$, rather than $\mathbf{z}_t$ in *LK-SDE*, $g_{\theta_1}(\mathbf{s}_t)$ as the diffusion coefficient matrix which is *diagonal and shared* by our *LK-SDE*. Therefore, the bicycle model could be expressed as $\mathbf{s}_{t+1} = h(\mathbf{s}_t, \pi) + g_{\theta_1}(\mathbf{s}_t)\Delta W_t$, where the drift function $h(\mathbf{s}_t, \pi)$ is shown in the following Eq. (4):

$$\begin{aligned}
x_{t+1} &= x_t + \delta \cdot v_t \cos(\psi_t + \beta(u_2)), \\
y_{t+1} &= y_t + \delta \cdot v_t \sin(\psi_t + \beta(u_2)), \\
v_{t+1} &= v_t + \delta \cdot u_1 \\
\psi_{t+1} &= \psi_t + \delta \cdot \frac{v_t}{l_r} \sin(\beta(u_2)), \\
\beta(u_2) &= \arctan\left(\tan(u_2)\frac{l_r}{l_f + l_r}\right), \\
(u_1, u_2) &= \pi(\mathbf{s}_t),
\end{aligned} \tag{4}$$

where the state vector $\mathbf{s}_t = (x_t, y_t, v_t, \psi_t)$ represents the *latent representation* of lateral position, longitudinal position, velocity, and yaw angle, respectively. β is the slip angle, and l_f, l_r are the distances between the car center and the front, and rear axle, respectively. $\delta > 0$ is a small sampling period. The control inputs correspond to the acceleration u_1 and front wheel steering angle u_2. A detailed illustration of the model is shown in Fig. 4.

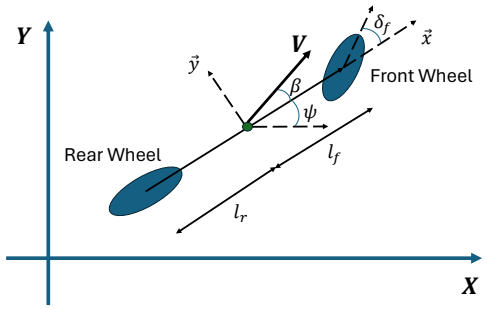

Fig. 4: Illustration of the bicycle model [24].

With the bicycle model dynamics $h(\cdot)$ fixed, we implement a *learnable feedback neural controller* π to generate the control inputs as $u_1, u_2 = \pi(\mathbf{s}_t)$, where π is learned from the output loss function as in Eq. (7). With a learned π, the bicycle model SDE in Eq. (4) can be viewed as an autonomous system evolving with the dynamics $h(\cdot, \pi), g_{\theta_1}(\cdot)$ within the latent space representation, and providing kinematics guidance for our *LK-SDE*.

2) Bicycle Model Guided LK-SDE: For our *LK-SDE*, the inputs for the neural network drift function $f_{\theta_0}(\cdot)$ are the latent states from the last time step, the global semantics, and local contexts. Following the transition function as shown in Eq. (5), our *LK-SDE* will generate the kinematics-aware vectors $\mathbf{z}$ for every time step in the latent space of the VAE:

$$\mathbf{z}_{t+1} = f_{\theta_0}(\mathbf{z}_t, \mathbf{sem}, \mathbf{ctx}_t) + g_{\theta_1}(\mathbf{z}_t)\Delta W_t. \tag{5}$$

To embed the kinematic knowledge from the bicycle model SDE to *LK-SDE*, we minimize the KL divergence between the solutions (rolled latent trajectories) $(\mathbf{z}_t, \mathbf{s}_t), t \in [0, \cdots, T]$ of two SDEs (similar to the Eq. 10 in torchsde [22] and the Appendix B in [25]):

$$L_{kin} = \sum_{t=0}^{T} \mathcal{KL}(\mathbf{z}_t || \mathbf{s}_t) = \sum_{t=0}^{T} \mathbb{E}_{\Delta W_t} \left[\frac{1}{2} \left\| \left(g_{\theta_1}^{-1} \left(f_{\theta_0}(\mathbf{z}_t, \mathbf{sem}, \mathbf{ctx}_t) - h(\mathbf{s_t}, \pi) \right) \right) \right\|_2^2 \right], \tag{6}$$

where T is the time length of the prediction and generation task. Same as the previous definition, g_{θ_1} is the diagonal and shared diffusion coefficient matrix in our *LK-SDE* and bicycle model SDE. Basically, we enforce the neural drift function $f_{\theta_0}(\mathbf{z}_t, \mathbf{sem}, \mathbf{ctx}_t)$ to get close to the bicycle model $h(\mathbf{z}_t, \pi)$ for kinematics knowledge, as shown in Fig. 3. In the training, we compute and minimize this loss function for every batch of samples.

Besides the kinematics loss function shown above, the *LK-SDE* is also optimized by the output loss function as in Eq. (7), by learning directly from the dataset. Therefore, two gradient backpropagations with kinematic knowledge and data knowledge jointly improve the learning performance of *LK-SDE* for more accurate, physically realistic, and controllable trajectory prediction and generation.

Discussions: We summarize several key points regarding our design of *LK-SDE* in the following.

- *Choice of SDE:* Rather than using other dynamical models such as ordinary differential equation (ODE), we choose SDE-based latent space, as inspired by the Gaussian-distribution-based latent space in [14], [25]. This is because SDE is able to encode stochasticity in the model (the random variable at a specific timestamp of SDE follows a Gaussian distribution), which is beneficial to effectively learning real-world data with random noises.
- *Going beyond a single bicycle-model SDE in the latent space:* This is due to two reasons: 1) Real-world data can exhibit a wide range of behaviors and uncertainties that a single over-simplified and fixed bicycle model may not be able to account for. Introducing variability in the latent space dynamics, such as using a more flexible model or allowing parameters to change over time, can enhance the model's ability to capture diverse patterns and adapt to different scenarios. This is why we use the bicycle model to serve as a soft constraint (loss function) for the *LK-SDE*. 2) The bicycle model solely considers the latent state, leading to a loss of information regarding global semantics and per-step local context, which hinders the model's learning capability.
- *Choice of a learnable π in the bicycle model for guidance:* This design aims to introduce variability and flexibility into the latent space dynamics, addressing the limitations of the naive and over-simplified bicycle model when handling real-world data.

D. Decoder for Output and the Optimization Pipeline

Decoder: Decoder D is designed as a fully-connected neural network. The decoder network D takes the latent vectors $(\mathbf{z}_0, \cdots, \mathbf{z}_T)$ produced by *LK-SDE* and outputs the waypoints of trajectory prediction and generation as

$$\hat{o}_{i:i+T} = D(\mathbf{z}_0, \cdots, \mathbf{z}_T).$$

For the output waypoints, we reduce the following loss in Eq. (7) to minimize the distance between output trajectories $\hat{o}$ and the ground truth o collected in the real world:

$$L_{pred}(o_i, \hat{o_i}) = \begin{cases} 0.5(o_i - \hat{o_i})^2 & \text{if } \|o_i - \hat{o_i}\| < 1 \\ \|o_i - \hat{o_i}\| - 0.5 & \text{otherwise} \end{cases} \tag{7}$$

As a global loss function, the gradient from Eq. (7) will backpropagate to every learnable sub-module including the GCN feature extractor, individual encoders, learnable controller in the bicycle model, *LK-SDE* and the decoder.

Optimization Pipeline: Overall, the training process of our approach is shown in Algorithm 1. We optimize and balance several loss functions to regularize the latent space and generate the final trajectories. We update the feature extractor and initial state encoder by the VAE regularization loss as shown in Eq. (3). The *LK-SDE* is optimized to embed the kinematic knowledge into latent vectors by L_{kin} in Eq. (6), and output loss function in Eq. 7 optimizes all components.

Limitations: The computation complexity of our approach is higher than that of existing approaches because our dual SDE design in the latent space involves optimization and calculation of both kinematic and data-driven models. This may limit the scalability to more complex kinematics models.

IV. Experiments

we conduct extensive experiments to evaluate the proposed methods. The experimental settings are introduced in Sec. IV-A. We demonstrate that our methods can generate physically realistic and more controllable augmented trajectories than baselines via visualized and statistical comparisons in Sec. IV-B. In Sec. IV-C, we further demonstrate the prediction accuracy of our approach and its ability to estimate unobservable variables with kinematic latent space.

A. Experiment Settings

We train our model on the Argoverse motion forecasting dataset [26] and evaluate the prediction performance on its validation set. The benchmark has more than 30K scenarios collected in Pittsburgh and Miami. Each scenario has a graph of the road map and trajectories of agents sampled at 10 Hz. In the motion generation and prediction tasks, we use the first 2 seconds of trajectories as input and generate the subsequent 3-second trajectories.

B. Physically Realistic and Controllable Trajectory Generation

As a generative model, our approach can generate diverse trajectories by tuning the latent space of our LK-SDE. We compare our methods with the learning-based generative model TAE [15] and the bicycle-model-based DKM [19]. We measure the metrics of jerk violation rate and the Wasserstein distance of distribution of acceleration to evaluate the physical realism of generated trajectories.

The jerk is the rate of change of an object's acceleration over time (the definition is in Eq. (8) below) and the magnitude of jerk is commonly used to represent the smoothness of trajectories [27], [28]. According to [27], the jerk threshold for discomfort presents about $0.3\ m/s^3$, ranging up to $0.9\ m/s^3$. In our work, we consider trajectories with jerks exceeding $0.9\ m/s^3$ as violations.

$$\mathbf{j}(t) = \frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t} = \frac{\mathrm{d}^2\mathbf{v}(t)}{\mathrm{d}t^2} = \frac{\mathrm{d}^3\mathbf{x}(t)}{\mathrm{d}t^3}. \tag{8}$$

As shown in Table I, our *LK-SDE* based model has the lowest average jerk value and lowest jerk violation rate [19], indicating our methods can generate smoother and more realistic trajectories than DKM and TAE. Specifically, in Fig. 5, TAE generates trajectories with sparse and unstable jerks – about 26% trajectories are with jerk magnitude above the discomfort threshold ($0.9\ m/s^3$), showing the difficulty of physically realistic trajectory generation by pure deep learning (DL)-based methods. For DKM, 8.7% of generated trajectories have higher jerk values than the threshold. **Our *LK-SDE* based model generates the smoothest and physically realistic motions** with only a 5.0% violation rate.

We also evaluate the realism of generated motion by measuring the acceleration distribution in the thrid column of Table I. Previous work [29] shows that the forward acceleration of vehicles can be precisely described by a generalized Pareto distribution by analyzing over 100 million real-world realistic data points from more than 100,000 kilometers. We evaluate the Wasserstein distance between the motion accelerations of generated trajectories and the reference Pareto distribution. Our approach has the smallest Wasserstein distance, indicating again that **our approach can generate the most physically realistic trajectories**.

TABLE I: Comparison between our *LK-SDE* based method and baselines including TAE and DKM on the smoothness and physical realism of generated trajectories.

Model \ Metrics	Average Jerk	Jerk Violation Rate	Acc. Wasserstein Distance to Ref.
TAE (DL-based) [15]	0.64	26%	2.20
DKM (Model-based) [19]	0.43	8.7%	0.52
LK-SDE (Ours)	**0.40**	**5.0%**	**0.45**

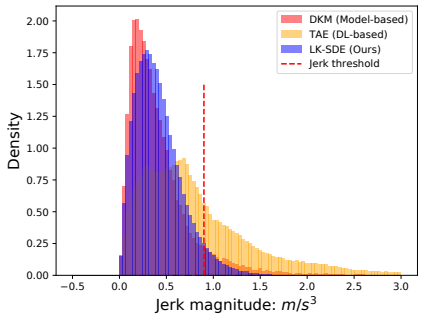

Fig. 5: The distribution of the jerk magnitude of different generative methods. The red dashed line represents the discomfort threshold for jerk value. We notice that our proposed *LK-SDE* can augment the smoothest trajectories.

Fig. 6 illustrates a few concrete examples. The top row shows that our *LK-SDE* model can accurately augment realistic vehicle motions in a physically feasible and controllable manner. We tune the initial states for lateral direction, longitudinal direction, and yaw angle, respectively, which can generate the corresponding diverse trajectories. We find that many trajectories generated by the baseline TAE method (bottom row in Fig. 6) have unrealistic and physically infeasible motions such as sharp turns and sudden offsets.

C. Accurate Prediction with Physics-informed Latent Space

The accuracy of trajectory prediction is a suitable metric to measure the ability to learn the representation of the

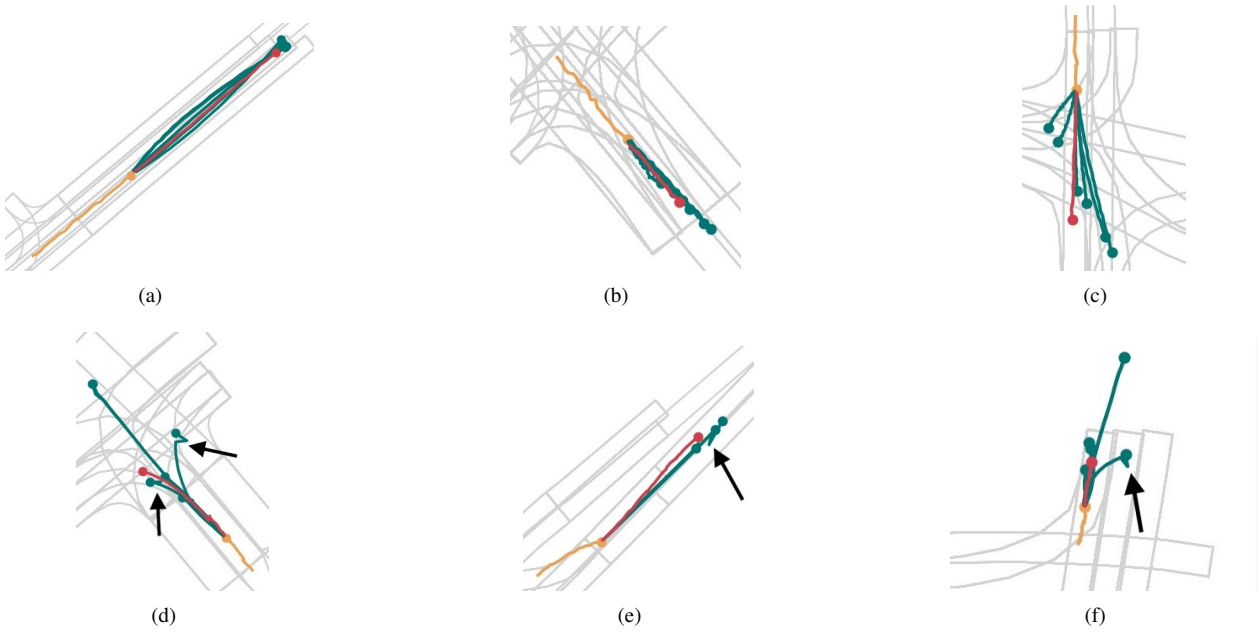

Fig. 6: Visualization of the augmented trajectories. The top row shows the trajectories generated by our *LK-SDE* model. We manually tune the initial values in the latent space (lateral for (a), longitudinal for (b), and yaw angle for (c)). The bottom row shows trajectories generated by the pure deep learning method TAE, by tuning its behavior latent space (lateral for (d), longitudinal for (e), random for (f)). In all the subplots, the orange lines are history trajectories, the red lines are ground truth for future motion, green lines are augmented trajectories. The black arrows point to the parts of trajectories that are obviously physically infeasible or unrealistic by TAE. The generated trajectories in (a)(b)(c) visually align with the physical knowledge from the bicycle model, showing the effectiveness of our *LK-SDE* design.

environment and generate realistic trajectories. Generally, the average performance of trajectory prediction or regression is measured by the average displacement error (ADE, defined as the average of the root mean squared error between the predicted waypoints and the ground-truth trajectory waypoints) and the final displacement error (FDE, defined as the root mean squared error between the last predicted waypoint and the last ground-truth trajectory waypoint). We compare our methods with the DL-based GRIP++ [30], LaneGCN [7], and TPCN [8], domain knowledge aware generative methods TAE [15], and bicycle-model-based DKM [19]. The results in Table II show that our *LK-SDE* model outperforms GRIP++, DKM, and TAE, and achieves close performance to the state-of-the-art LaneGCN and TPCN in prediction accuracy (note that LaneGCN, TPCN, and GRIP++ do not have trajectory generation capability).

TABLE II: Trajectory prediction comparison between our *LK-SDE* based method and baselines.

Metrics / Model	ADE	FDE
GRIP++ [30]	1.77	3.91
LaneGCN [7]	1.35	2.96
TPCN [8]	1.34	2.95
TAE [15]	1.42	3.08
DKM [19]	1.46	3.14
LK-SDE (Ours)	1.39	2.98

Besides, our kinematics-aware latent space can estimate some variables that are crucial for understanding the target vehicle's behavior but cannot be observed directly by perception or prediction modules. In our design, the bicycle-model-based LK-SDE can estimate the normalized steering angles ($u2$) and slip angles (β) without explicit training on ground truth data. Fig. 7 shows the distribution of normalized steering angles on the Argoverse dataset, estimated by the kinematics-aware latent space. The normalized steering angle and its distribution can help us discern the steering directions as well as the sharpness of the turns. For instance, when

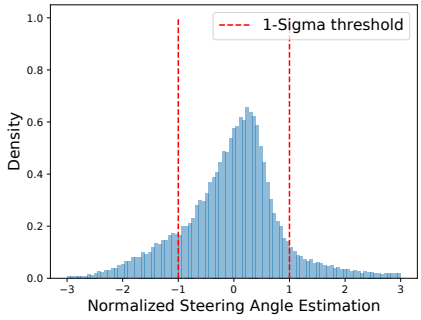

Fig. 7: The distribution of the normalized steering angles estimated in the latent space. The red dashed line represents the one-sigma threshold. Our kinematics-aware method can estimate unobservable variables such as steering angle, which is useful for understanding target vehicles' behavior.

the estimated steering angle of a vehicle is larger than a threshold (such as the one-sigma range in Fig. 7), it suggests the possibility of an aggressive turning maneuver.

Combining the results from trajectory generation and augmentation, we can conclude that our latent kinematics-aware SDE can learn the representation of trajectories and environments more effectively than the model-based methods and generate more physically realistic and controllable motions than the DL-based generative models. In addition to the average accuracy, our methods can also **give detailed and accurate kinematics prediction (e.g., steering angle) along with the waypoints**, which provide more explainable information for safety-critical decision-making.

V. Conclusion

In this work, we propose a vehicle motion generator with the latent kinematics-aware stochastic differential equation (*LK-SDE*). We embed the physics knowledge into the latent space of a VAE by the dual SDE design. The method can bridge the high-dimensional features from the environment and the low-dimensional kinematics to generate fine-grained, physically realistic, and controllable trajectories, and to provide accurate prediction of unobservable physical variables.

Acknowledgment

We graciously acknowledge support from National Science Foundation grants 2324936, 2328973 and 2328032. This work is also supported by the grant EP/Y002644/1 under the EPSRC ECR International Collaboration Grants program, funded by the International Science Partnerships Fund (ISPF) and the UK Research and Innovation.

References

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

[2] X. Liu, R. Jiao, Y. Wang, Y. Han, B. Zheng, and Q. Zhu, "Safety-assured speculative planning with adaptive prediction," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[3] X. Liu, C. Huang, Y. Wang, B. Zheng, and Q. Zhu, "Physics-aware safety-assured design of hierarchical neural network based planner," in *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2022, pp. 137–146.

[4] X. Liu, R. Jiao, B. Zheng, D. Liang, and Q. Zhu, "Safety-driven interactive planning for neural network-based lane changing," in *Proceedings of the 28th Asia and South Pacific design automation conference*, 2023, pp. 39–45.

[5] W. Ding, C. Xu, M. Arief, H. Lin, B. Li, and D. Zhao, "A survey on safety-critical driving scenario generation—a methodological perspective," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[6] J. Cai, W. Deng, H. Guang, Y. Wang, J. Li, and J. Ding, "A survey on data-driven scenario generation for automated vehicle testing," *Machines*, vol. 10, no. 11, p. 1101, 2022.

[7] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *ECCV*. Springer, 2020, pp. 541–556.

[8] M. Ye, T. Cao, and Q. Chen, "Tpcn: Temporal point cloud networks for motion forecasting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 318–11 327.

[9] Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani, "Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9813–9823.

[10] Z.-H. Yin, L. Sun, L. Sun, M. Tomizuka, and W. Zhan, "Diverse critical interaction generation for planning and planner evaluation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7036–7043.

[11] W. Ding, M. Xu, and D. Zhao, "Cmts: A conditional multiple trajectory synthesizer for generating safety-critical driving scenarios," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4314–4321.

[12] W. Ding, H. Lin, B. Li, and D. Zhao, "Causalaf: causal autoregressive flow for safety-critical driving scenario generation," in *Conference on Robot Learning*. PMLR, 2023, pp. 812–823.

[13] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany, "Generating useful accident-prone driving scenarios via a learned traffic prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 305–17 315.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[15] R. Jiao, X. Liu, B. Zheng, D. Liang, and Q. Zhu, "Tae: A semi-supervised controllable behavior-aware trajectory generator and predictor," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 12 534–12 541.

[16] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, *et al.*, "Tnt: Target-driven trajectory prediction," in *Conference on Robot Learning*. PMLR, 2021, pp. 895–904.

[17] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "Wayformer: Motion forecasting via simple & efficient attention networks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2980–2987.

[18] C. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, D. Anguelov, *et al.*, "Motiondiffuser: Controllable multi-agent motion prediction using diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9644–9653.

[19] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 563–10 569.

[20] P. E. Kloeden, E. Platen, P. E. Kloeden, and E. Platen, *Stochastic differential equations*. Springer, 1992.

[21] E. Cinlar, *Introduction to stochastic processes*. Courier Corporation, 2013.

[22] X. Li, T.-K. L. Wong, R. T. Chen, and D. Duvenaud, "Scalable gradients for stochastic differential equations," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 3870–3882.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 812–818.

[25] P. Kidger, J. Foster, X. C. Li, and T. Lyons, "Efficient and accurate gradients for neural sdes," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 747–18 761, 2021.

[26] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.

[27] I. Bae, J. Moon, J. Jhung, H. Suk, T. Kim, H. Park, J. Cha, J. Kim, D. Kim, and S. Kim, "Self-driving like a human driver instead of a robocar: Personalized comfortable driving experience for autonomous vehicles," *Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems*, 2020.

[28] A. Scamarcio, P. Gruber, S. De Pinto, and A. Sorniotti, "Anti-jerk controllers for automotive applications: A review," *Annual Reviews in Control*, vol. 50, pp. 174–189, 2020.

[29] R. Liu, X. Zhao, X. Zhu, and J. Ma, "Statistical characteristics of driver acceleration behaviour and its probability model," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 236, no. 2-3, pp. 395–406, 2022.

[30] X. Li, X. Ying, and M. C. Chuah, "Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving," *arXiv preprint arXiv:1907.07792*, 2019.