Transportation Research Part E 192 (2024) 103799

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Physics-informed neural network for cross-dynamics vehicle trajectory stitching

Keke Long [a], Xiaowei Shi [b,*], Xiaopeng Li [a,*]

[a] Department of Civil and Environmental Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA
[b] Department of Civil and Environmental Engineering, University of Wisconsin-Milwaukee, Milwaukee, WI 53211, USA

ARTICLE INFO

Keywords:
Physics-informed Neural Network
Trajectory Reconstruction
Vehicle Trajectory Dataset
Extrapolation

ABSTRACT

High-accuracy long-coverage vehicle trajectory data can benefit the investigations of various traffic phenomena. However, existing datasets frequently contain broken trajectories due to sensing limitations, which impedes a thorough understanding of traffic. To address this issue, this paper proposes a Physics-Informed Neural Network (PINN)-based method for stitching broken trajectories. The proposed PINN-based method enhances traditional neural networks by integrating physics priors, including vehicle kinematics and boundary conditions, aiming to provide information beyond training domain and regularization, thus increasing method accuracy and extrapolation ability for cross-dynamics scenarios (e.g., extrapolating from low-speed training data to reconstruct high-speed trajectories). Two publicly available vehicle trajectory datasets, NGSIM and HighSIM, were adopted to validate the proposed PINN-based method, and four biased training scenarios were designed to assess the PINN-based method's extrapolation ability. Results indicate that the PINN-based method demonstrated superior performance regarding trajectory stitching accuracy and consistency compared to benchmark models. The dataset processed using our proposed PINN-based method has been made publicly available online to support the traffic research community. Additionally, this PINN-based approach can be applied to a broader range of scenarios that include physics-based priors.

1. Introduction

Vehicle trajectories, which represent the positions of a stream of vehicles over time along a guideway, offer valuable insights for various traffic-related studies, including traffic flow theory, simulation modeling, safety measures, and traffic management. Existing vehicle trajectory data collection methods could be categorized into two main types (Kim and Cao, 2010): vehicle-based methods (Anuar and Cetin, 2017; Coifman et al., 2016; Victor, 2014; Zhao et al., 2017) and video-based methods (Ke et al., 2019; Kim et al., 2019; Xu et al., 2017; Zhao and Li, 2019). The vehicle-based methods suggest collecting vehicle trajectory data by probe vehicles. Vehicles outfitted with positional and distance measuring sensors, such as Lidar, Radar, and GPS, navigate the test road segment, allowing for the collection of trajectories from these probe vehicles and the vehicles around them. The drawback of the vehicle-based methods is obvious. As only the trajectories of the probe vehicles and their surrounding vehicles are gathered, the overall data penetration rate relative to the total traffic volume tends to be quite limited.

On the other side, the swift advancement of aerial video recording technologies, such as unmanned aerial vehicles equipped with

* Corresponding authors.
E-mail addresses: tomshi@uwm.edu (X. Shi), xli2485@wisc.edu (X. Li).

https://doi.org/10.1016/j.tre.2024.103799
Received 19 May 2024; Received in revised form 23 September 2024; Accepted 25 September 2024
Available online 16 October 2024

high-definition cameras, is enhancing the appeal of video-based methods. These approaches offer benefits like scalability, flexibility, cost-effectiveness, and impartiality (Kim and Cao, 2010). Therefore, the collection of video-based trajectory data has recently garnered significant interest from researchers in both industry and academia (Berghaus et al., 2024; Chen et al., 2021; Long et al., 2024b; Shi et al., 2021). Despite the advantages offered by video-based data collection technologies, it is important to highlight two fundamental issues that affect the datasets: detection errors and limited coverage, as detailed below.

Detection errors can be categorized based on their origin into two types: source errors and extraction errors. Source errors occur due to occasional losses in video feed, often a result of recording from aerial platforms. For instance, the target sites might be obstructed by surrounding structures such as bridges, traffic signals, billboards, or buildings, thus vehicle motions in the blocked areas are lost. As circled in purple in Fig. 1, a bridge across the recorded road segment leads to the loss of the trajectories of all vehicles passing beneath it.

Extraction errors stem from the methodologies used for trajectory extraction. To extract trajectory data from video sources, a variety of methods have been developed to track vehicle movements within videos. Since videos were recorded aerial at high altitudes, vehicles in the videos are small and usually with only a few pixels. Thus, reliable vehicle tracking is an extremely challenging problem, which falls into the computer vision field and has attracted intensive studies in the past few years (Jazayeri et al., 2011; Wang et al., 2008; Zhang et al., 2007). Despite the development of numerous high-performance methods aimed at resolving this issue, no technique can consistently ensure superior detection rates. External factors such as weather conditions, lighting, wind, and camera angles can also adversely affect the quality of the recorded video and consequently, the detection rates. Therefore, missing detections during the trajectory extraction process are often unavoidable. Once a missing detection happens, the original continuous long trajectory gets fragmented into shorter segments at the points of missed detection, compromising the dataset's quality. As circled in black in Fig. 1, the vehicle sizes in the video diminish as they approach the road segment's end due to camera angles. This reduction in size leads to lower detection rates, resulting in the extracted trajectories being segmented into smaller sections downstream.

To address the issue of detection errors, studies on vehicle trajectory data post-processing were conducted (Coifman et al., 2016; Lee and Krumm, 2011; Punzo, 2009; Xin, 2008). Interested readers can refer to Lee and Krumm (Lee and Krumm, 2011) for a detailed review of this research stream. It is noteworthy that this topic has gained attention in recent years with the advent of aerial video recording technologies, but most existing methods rely on simple movement trends (Zhang and Jin, 2019), such as using constant velocity (Kim et al., 2019) or interpolation (Raju et al., 2022; Tong et al., 2017; Zhang and Jin, 2019; Zheng et al., 2023). These simplifications may yield limited quality results, failing to capture real-world driving behaviors (e.g., car-following behavior). One research (Sazara et al., 2017) extended broken trajectories using a car-following model. However, they simply connected the extended trajectory to another one, which inevitably creates abrupt changes in vehicle motions at the connection point. Additionally, the adopted car-following model fails to depict driver heterogeneity in traffic (Durrani et al., 2016; Punzo and Montanino, 2020). In summary, the accuracy of these physics model-based trajectory stitching methods is compromised by oversimplified assumptions of vehicle motions and the neglect of driver heterogeneity.

One alternative solution to counter the physics model-based methods' low accuracy is to use learning-based methods, which can extract driving behaviors and heterogeneity from data. There are a few studies investigating the trajectory stitching problem using learning-based methods. For example, (Hepburn and Montana, 2024) used reinforcement learning, and (Xiao et al., 2022) used transfer learning methods to address issues related to broken trajectories. Despite the excellent predictability of learning-based methods, these methods do not incorporate physics information and lack interpretability. In addition, learning-based methods heavily rely on large datasets and lack the extrapolation ability to domain with sparse training data (Fang et al., 2023; Lin et al., 2023;

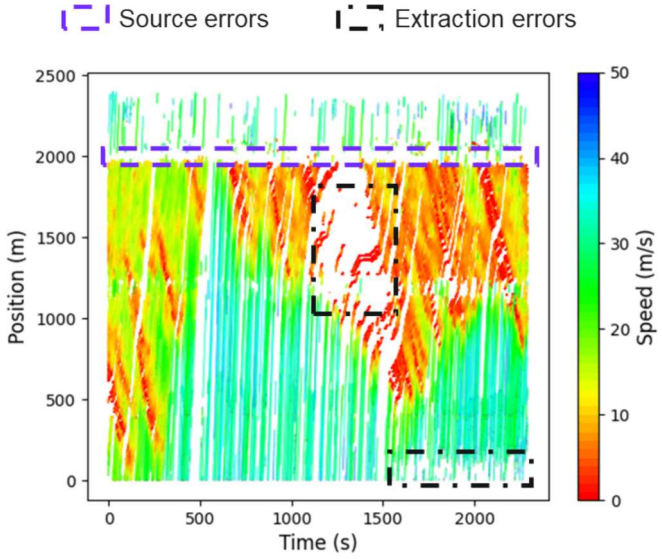

Fig. 1. Source error and extraction error in HighSIM origin data (Shi et al., 2021).

Liu et al., 2023; Wang et al., 2023). It typically requires substantial amounts of data across all situations to be effective, which may lead to irrational results (e.g., abnormally large accelerations, etc.) in unseen scenarios where extensive high-quality data is not easily accessible.

To overcome the lack of interpretability and extrapolation ability of learning-based methods, a promising solution is the physics-informed Neural Network (PINN) method: Neural networks are encoded with physics models to enhance interpretability and predictability (Karniadakis et al., 2021; Long et al., 2024a; Raissi and Karniadakis, 2018). Experiments have demonstrated that PINN models outperform purely data-driven models due to their enhanced interpretability and predictability. By integrating physical laws into their structure, PINNs align model outputs with established physical expectations, making insights more comprehensible to human experts. Additionally, these physical laws prevent overfitting on small datasets, enabling PINNs to forecast outcomes accurately even when training data is limited. (Liu et al., 2021; Mo et al., 2021). However, trajectory stitching is a relatively overlooked problem in literature, and this excellent method has not been noticed in the literature.

Overall, physics model-based methods have oversimplified assumptions of vehicle motions and the neglect of driver heterogeneity. Learning-based methods account for factors overlooked by physics model-based methods, but they lack interpretability and exhibit unstable performance in unseen domains, as illustrated in the upper subfigure of Fig. 2. PINN serves as a new method that incorporates the advantages of the two mentioned methods but overcomes the drawbacks. As illustrated in the lower subfigure of Fig. 2, the physics prior offers a regularization scheme during the training process, improving the extrapolation ability on unseen data. Due to this, our study applies the PINN model to the vehicle trajectory stitching problem. To illustrate the performance of the proposed method, we validated it on both NGSIM and HighSIM datasets. The results showed that the proposed method outperformed several benchmark methods in both trajectory reconstruction accuracy and trajectory consistency aspects. Moreover, our method demonstrated strong predictive ability in domains not covered by the training sample, proving that physics factors significantly enhance the model's extrapolation ability.

The contributions of this paper to the literature are threefold:

1. This paper proposed a PINN-based method tailored to address the trajectory stitching problem. This method incorporates the advantages of the physics model-based and learning-based methods but overcomes the drawbacks. Note that this method is not limited to a specific application but is a versatile framework that can be adapted across various domains.
2. The proposed PINN-based method has fairly good extrapolation ability, making it suitable for diverse traffic dynamics and ensuring robust performance across varying scenarios.
3. The dataset processed using the proposed PINN-based method referred to as HIGH-SIM, has been published via shared link of both the Federal Highway Administration, U.S. Department of Transportation (https://highways.dot.gov/) and Connected and Autonomous Transportation Systems Lab, University of Wisconsin-Madison (https://github.com/CATS-Lab) for public use.

The structure of this paper is organized as follows: Section 2 outlines the trajectory stitching problem that is the focus of this investigation. Section 3 details the proposed PINN method for vehicle trajectory stitching. Section 4 presents a series of numerical experiments designed to evaluate the effectiveness of the proposed method. Finally, Section 5 provides conclusions, discusses the limitations of this research and explores future improvements.

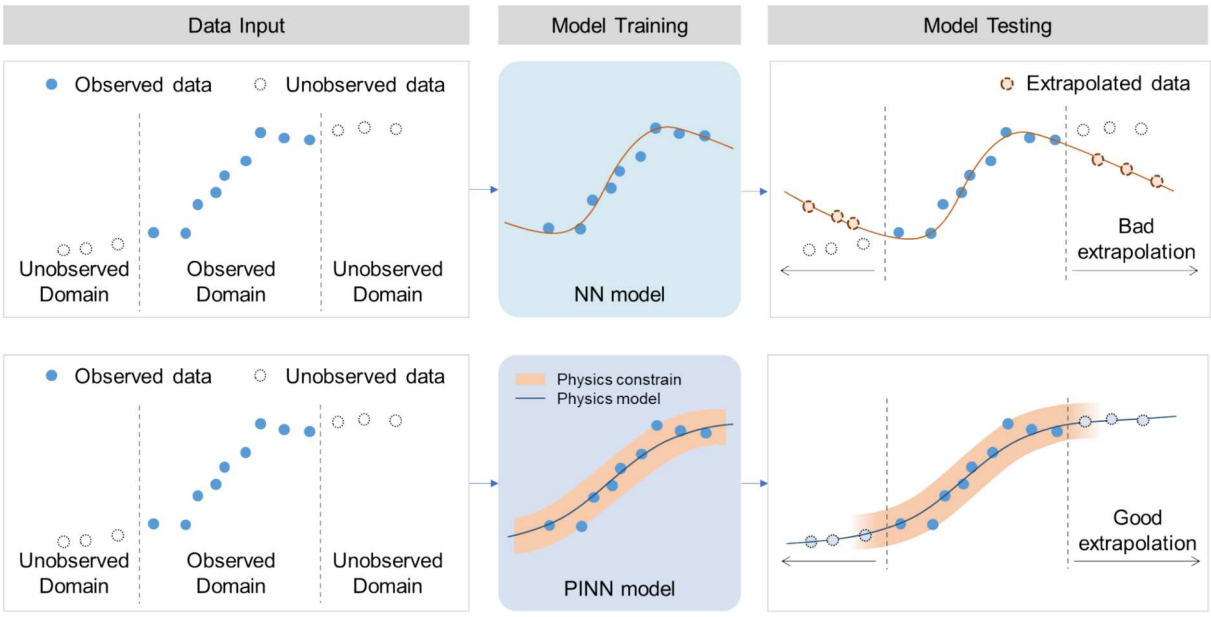

Fig. 2. General description of the extrapolation ability of PINN.

2. Problem statement

In this section, we first define the different kinds of broken trajectories that need to be stitched, then we introduce the trajectory stitching problem. Table 1 includes the key notation in this research.

2.1. Broken vehicle trajectory

Let $\mathcal{N}$ denote the set of vehicle trajectories (hereafter referred to as 'trajectory') in the investigated dataset. Each trajectory is labeled as $n \in \mathcal{N}$. The trajectory data are captured in a spatial range $[0, T]$ within a continuous time period. In practice, data are typically available only at discrete time points, thus the time period is discretized into a set of time points $\mathcal{T} := \{0, 1, 2, \cdots, T\}$ with a certain time interval Δ. For each trajectory, the beginning time point is $t_n^b \in \mathcal{T}$ and the ending time point is $t_n^e \in \mathcal{T}$, thus the trajectory is defined as $[d_{nt}]_{\forall t \in [t_n^b, t_n^b+\Delta, \cdots, t_n^e]}$ with time point array $t \in [t_n^b, t_n^b+\Delta, \cdots, t_n^e] \subseteq \mathcal{T}$ denoting the consecutive time points. Due to the aforementioned issues, tt is expected that a number of trajectories in dataset $\mathcal{N}$ only capture a portion of the subject vehicles' movements. These trajectories, referred as *broken trajectories*, can be identified if they satisfy either of the following conditions: (1) if $t_n^b > 0$ and $d_{t_n^b} > 0, \forall n \in \mathcal{N}$; or (2) if $t_n^e < T$ and $d_{t_n^e} < D, \forall n \in \mathcal{N}$. The trajectory that is complemented by the proposed method is called stitched trajectory.

All incomplete trajectories from dataset $\mathcal{N}$ are collected and represented as a subset, denoted by $\mathcal{N}^b \subseteq \mathcal{N}$. To analyze the missing segments of each incomplete trajectory, the broken trajectory dataset $\mathcal{N}^b$ is classified into three subsets, $\mathcal{N}_1^b, \mathcal{N}_2^b, \mathcal{N}_{12}^b$. Broken trajectories that only satisfy condition (1) but not condition (2) are stored in $\mathcal{N}_1^b$, which means that these trajectories are broken at the origin side (i.e., a segment before trajectory n is missed). Broken trajectories that only satisfy condition (2) but not condition (1) are stored in $\mathcal{N}_2^b$, which means that these trajectories are broken at the end side (i.e., a segment after trajectory n is missed). Broken trajectories that satisfy both conditions (1) and (2) are stored in $\mathcal{N}_{12}^b$, which means that these trajectories are broken at both sides. Therefore, $\{\mathcal{N}_1^b \cup \mathcal{N}_2^b\} = \mathcal{N}^b$, and $\{\mathcal{N}_1^b \cap \mathcal{N}_2^b\} = \{\mathcal{N}_{12}^b\}$. For the specific example shown in Fig. 3, $\mathcal{N} = \{1,2,\cdots,8\}$, $\mathcal{N}^b = \{2,3,5,6,7\}$, $\mathcal{N}_1^b = \{3,6,7\}$, $\mathcal{N}_2^b = \{2,5,6\}$, $\mathcal{N}_{12}^b = \{6\}$.

2.2. Trajectory stitching.

$N \in \mathcal{N}^b$ is the index of the target broken trajectory that needs to be stitched to be processed. The target broken trajectory is denoted as $[d_{Nt}]_{\forall t \in [t_N^b, t_N^e]}$, where $[t_N^b, t_N^e]$ is the set of consecutive time stamps separated by a customized unit time interval Δ, t_N^b is the beginning time point of the target trajectory N, and t_N^e is the ending time point of the target trajectory N, shown in Fig. 4.

Since we do not initially know the correspondence between trajectories, the surrounding trajectory that may be connected to the target trajectory is denoted as $n' \in \mathcal{N}$. Assuming that the trajectories N and n' belong to the same vehicle, the real missing trajectory of

Table 1
Key notation.

Notation	Description
$\mathcal{T}$	The researched time period
T	Maximum of researched range of time point $T \in \mathcal{T}$
t	Index of time $t \in \mathcal{T}$
Δ	A customized unit time interval
t_n^b	The beginning time point of historical trajectories before the broken part when predicting the broken trajectory N
t_n^e	The ending time point of historical trajectories before the broken part, also the beginning time point of stitched trajectory N
t_N^s	The ending time point of the stitched trajectory
$\mathcal{N}$	Set of vehicle trajectories
$\mathcal{N}^b$	Set of broken vehicle trajectories,$\mathcal{N}^b \subseteq \mathcal{N}$
$\mathcal{N}_1^b, \mathcal{N}_2^b, \mathcal{N}_{12}^b$	Subset of broken vehicle trajectories,$\mathcal{N}_1^b, \mathcal{N}_2^b, \mathcal{N}_{12}^b \subseteq \mathcal{N}^b$
n	Index of trajectory.$n \in \mathcal{N}$
N	Index of the studied broken trajectory,$N \in \mathcal{N}$
$\mathcal{N}'$	The set of historical trajectories used for stitching the broken trajectory N, $\mathcal{N}' \subseteq \mathcal{N}$
d_{nt}	Position of vehicle n at time point t
D	Maximum of researched range of vehicle location
θ	Parameters of neural network model in PINN
λ	Parameters of physics model in PINN
Θ	Feasible domain of the neural network parameters θ
Λ	Feasible domain of the physics parameters λ
E_N	Reconstruction error of trajectory N
u_t	latent (hidden) solution of a nonlinear ordinary differential equation (ODE)
$O[\cdot]$	A nonlinear differential operator
$P[\cdot]$	A differential operator representing a physics rule
$\mathcal{I}$	Set of samples
i	Index of samples,$i \in \mathcal{I}$

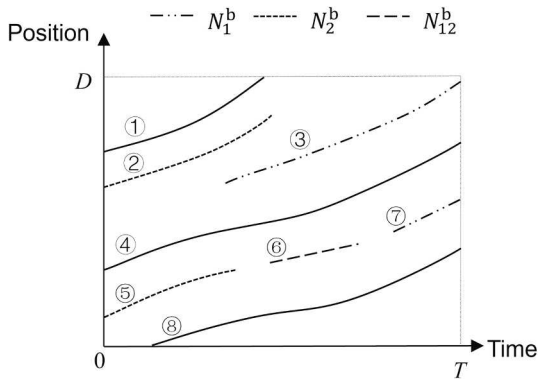

Fig. 3. Three kinds of broken trajectory.

trajectory N is $[d_{Nt}]_{\forall t\in[t_N^e,t_{n'}^b]}$, where $t_{n'}^b$ is the beginning time point of the target trajectory n'. The predicted stitched trajectory is $[\widehat{d}_{Nt}]_{t\in[t_N^e,t_N^s]}$, where t_N^s is the ending time point of the stitched trajectory. Our approach relies on matching the predicted trajectory with the surrounding trajectory, ensuring that the predicted trajectory closely approximates the actual trajectory, without abrupt changes occurring at the connection points with the end states. Therefore, we define the reconstruction error as E_N, during the reconstruction period $[t_N^e,t_{n'}^b]$, which is the difference between the reconstructed and actual positions of the vehicle within the predicted time frame.

$$E_N=\sum\nolimits_{t\in[t_N^e,t_{n'}^b]}(d_{Nt}-\widehat{d}_{Nt})^2\# \tag{1}$$

This paper aims to connect these broken trajectories considering physics prior knowledge and thus enhances the quality of the dataset. Our task is to train a model to reconstruct the stitched trajectory for trajectory N: $y_N:=f(\bullet|\theta,\lambda)$, θ and λ are parameters of the proposed model aimed at minimizing prediction errors:

$$\min_{\theta\in\Theta,\lambda\in\Lambda}E_N\# \tag{2}$$

$$s.t.[\widehat{d}_{Nt}]_{t\in[t_N^e,t_{n'}^b]}=f\left([d_{nt}]_{n\in\mathcal{N},t\in[t_n^b,t_n^e]}|\theta,\lambda\right)\# \tag{3}$$

Extrapolation ability refers to a model's capability to make accurate predictions on new, unseen data. This ability reflects the real-world applicability to adapt to and predict under different conditions. In this work, we assess the model's extrapolation ability by training and testing it across domains with different dynamic characteristics, termed 'cross-dynamic domains'. This approach tests the model's adaptability and predictive accuracy in scenarios that differ significantly from those seen during training.

3. PINN solution for ordinary differential equations

In this work, we consider a nonlinear ordinary differential equation (ODE) in a general form:

$$u_t+O[u_t]=0,t\in T\# \tag{4}$$

where u_t is the latent (hidden) solution, $O[\cdot]$ is a nonlinear differential operator, t is time. This setup covers a broad range of

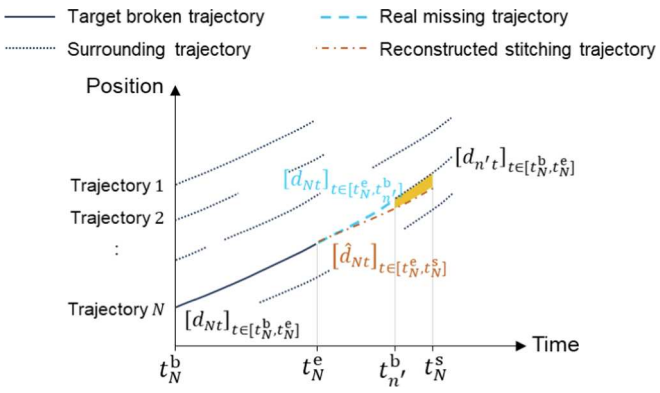

Fig. 4. Model input and output.

problems in transportation, including individual vehicle behavior and traffic flow. Our goal is to find a solution for the ODE (Raissi et al., 2019). Our understanding of the system frequently includes certain physics-based prior knowledge, such as conservation principles governing the observed data or established physics models, which can aid in finding a solution to Eq. (1). In the following discussion, we focus on how to ensure that the data-driven solution, typically provided by neural networks, is constrained and adheres to physics priors.

The first kind of physics priors is principles governing the observed data, denoted as:

$$u_t + P(u_t) = 0, t \in T\# \tag{5}$$

where P is a differential operator representing a physics rule. For instance, in vehicle trajectory stitching problems, this might be boundary constraints such as initial vehicle state; In traffic flow problems, the Lighthill-Whitham-Richards (LWR) model; In operations research, such as truck–drone hybrid delivery system, P refers to boundary conditions that govern the total inbound and outbound flows (She and Ouyang, 2024). These physics rules do not suffer from accuracy issues; they are essential conditions that solutions to the ODE must satisfy.

The second kind of physics prior is physics models $\widehat{O}(u_t|\lambda)$, which is an understanding and abstraction of $O[u_t]$. However, physics models often unavoidably entail a small error ϵ_t:

$$u_t + \widehat{O}(u_t|\lambda) = \epsilon_t, t \in T\# \tag{6}$$

where $\widehat{O}$ is a differential operator representing the ODE. $\lambda \in \Lambda$ represents the parameters of the physics model in a feasible domain Λ. Examples of physics models include the car-following models in vehicle longitudinal behavior prediction (Long et al., 2024a), and the user equilibrium assignment model in traffic flow prediction (Zhang et al., 2024). Both types of models are abstracted based on the understanding of physics laws and inherently contain some assumptions, hence the presence of error.

Denote the Physics-Informed Learning (PINN) model we designed (typically a neural network) as $f(u^i|\theta,\lambda)$, where θ,λ are the parameters of the PINN model and $i \in \mathscr{I}$ is the index of samples. Our objective is to make $f(\theta,\lambda)$ approximate $O[u]$ as closely as possible:

$$\min_{\theta\in\Theta,\lambda\in\Lambda} \sum\nolimits_{t\in\mathscr{T},i\in\mathscr{I}} \left(f\left(u_t^i|\theta,\lambda\right) - O\left[u_t^i\right]\right)^2\# \tag{7}$$

$$s.t.\ P\left(u_t^i|\lambda\right) = 0, t \in \mathscr{T}, i \in \mathscr{I}$$

The model is learned by minimizing the loss function $\mathscr{L}$ with physics-informed regularization:

$$L = L + w_1 R_1 + w_2 R_2\# \tag{8}$$

where L is the difference between the PINN model output and the real value, which is the same as the optimization goal in Eq. (4).

$$L = \sum\nolimits_{t\in\mathscr{T},i\in\mathscr{I}} \left(f\left(u_t^i|\theta,\lambda\right) - O\left[u_t^i\right]\right)^2\# \tag{9}$$

R_1 is a physics regularization term from the physics constraint, which is the error of the physics constraint:

$$R_1 = \sum\nolimits_{t\in\mathscr{T},i\in\mathscr{I}} \left(f\left(u_t^i|\theta,\lambda\right) - P\left(u_{t'}^i\right)\right)^2\# \tag{10}$$

Compared to traditional L^1 and L^2 regularization methods, which primarily reduce the weights in the neural network model, the physics constraint-based regularization term R_1 not only constrains the weights but also the bias of each neuron (Nabian and Meidani, 2020). Therefore, this regularization term provides stronger constraints, helping the model avoid overfitting in scenarios where data is erroneous or insufficient.

R_2 is a physics regularization term from the physics model, which is the difference between the PINN model output and the physics model:

$$R_2 = \sum\nolimits_{t\in\mathscr{T},i\in\mathscr{I}} \left(f\left(u_t^i|\theta,\lambda\right) - \widehat{O}\left(u_t^i|\lambda\right)\right)^2\# \tag{11}$$

The R_2 regularization term leverages the prior information contained in the physics model to enhance the interpretability of the

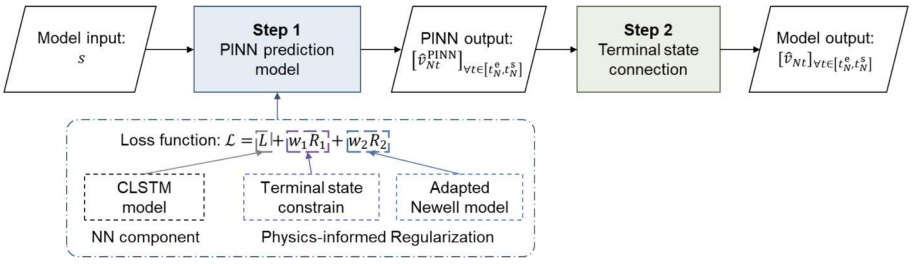

Fig. 5. The structure of the PINN trajectory stitching model in the training process.

PINN model. It also assists the model in maintaining strong performance in domains beyond the training set. This approach is analogous to data augmentation with the physics model, meaning in the absence of data for certain domains, the known physics model is used to generate data for those domains, thus supplementing the dataset.

4. Trajectory stitching methodology

This section introduces the ability of the PINN method to solve the trajectory stitching problem. The trajectory stitching process is shown in Fig. 5. It contains two steps; the first step is the proposed PINN method to get an initial prediction of the stitched trajectory. The second step is terminal state connection to modify the predicted stitched trajectory. Then, we will show a detailed description of the two sub-steps and their specific configurations.

4.1. Step 1 PINN prediction

The proposed PINN vehicle trajectory stitching method consists of two components: a neural network (NN) component and a physics component.

Given the sequential feature of the input, a Convolutional Long Short-Term Memory (CLSTM)-based model is adopted to learn features sequentially following a basic car-following rule that vehicles would follow their immediately preceding vehicles one by one from downstream to upstream. The CLSTM model integrates convolutional layers for abstracting trends and handling spatial dependencies within the data, enhancing the accuracy and robustness of predictions. The LSTM layers focus on temporal dependencies and event sequences, crucial for modeling car-following behaviors where vehicles sequentially follow each other (Lim et al., 2021; Zhu et al., 2021).

The model structure is shown in Fig. 6. It starts with an input layer that processes vehicle state data, followed by convolutional layers for initial feature extraction. LSTM layers then analyze these features to understand time-related changes in the traffic environment. Dropout layers interspersed within the LSTM layers prevent overfitting by omitting subsets of features, thereby improving the model's generalizability. The architecture concludes with dense layers that compile the learned features into predictions of vehicle behaviors, topped with a final dropout layer to reduce overfitting. This design leverages the strengths of both convolutional and LSTM networks, making the CLSTM model highly suitable for complex traffic scenarios. Note that other types of neural networks, like recurrent neural networks, can also be adopted.

The input of the NN component contains both the position and speed of the trajectory N and its $N-1$ preceding trajectories: $s^{\mathrm{NN}} := \left\{ [d_{nt}]_{n\in\mathscr{N},\forall t\in[t_N^e - T^b\delta, t_N^e]}, [v_{nt}]_{n\in\mathscr{N},\forall t\in[t_N^e - T^b\delta, t_N^e]} \right\}$, where $\mathscr{N} := \{1, 2, \cdots, N\} \subseteq \mathscr{N}$ is the set of trajectories considered when stitching the broken trajectory for vehicle N. T^b is the number of backward steps as the historical trajectory, $t_N^e - T^b\delta \geq t_N^b$. The output of the NN component is the predicted vehicle speed $\left[\widehat{v}_{Nt}^{\mathrm{NN}}\right]_{\forall t\in[t_N^e, t_N^s]}$.

The training error of the NN component is the discrepancy between the observed speed and that predicted by the NN component:

$$L = \sum\nolimits_{t\in[t_N^e, t_N^s]} \left(v_{Nt} - \widehat{v}_{Nt}^{\mathrm{NN}}\right)^2 \# \tag{12}$$

The physics component includes two main characteristics of vehicle trajectory: shockwave physics and kinematic constraints, as two physics priors.

The first part of physics prior is shockwave, a critical feature of congested traffic flow, which infers the behavior of a vehicle based on the trajectories of surrounding vehicles. We calibrated a shockwave-based car-following model as the physics model (Long et al., 2024b; Yao et al., 2023), which is characterized by a single parameter w denoting the wave speed, and thus $P = 1$. The model input is the historical trajectory information $s^{\mathrm{Phy}} = [v_{nt}, d_{nt}]_{n\in\mathscr{N},\forall t\in[t_N^e - T^b\delta, t_N^e]}, i \in \mathscr{I}$, and the model output is the predicted future speed $\widehat{y}^{\mathrm{Phy}} = \left[\widehat{v}_{Nt}^{\mathrm{Phy}}\right]_{\forall t\in[t_N^e, t_N^s]}, i \in \mathscr{I}$, given by:

$$\widehat{v}_{Nt}^{\mathrm{Phy}} = v_{nt - \frac{D_{iNnt}}{w}}, \forall t \in [t_N^e, t_N^s] \# \tag{13}$$

where D_{iNnt} represents the position distance between vehicle N and vehicle n at time t in sample i. This approach enables us to move

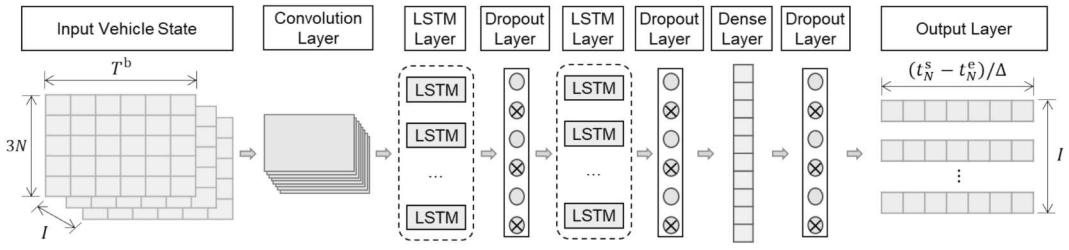

Fig. 6. The structure of the NN component: a CLSTM model.

beyond the limitations of the car-following model, which only considers the vehicle directly ahead. Our predictions can be based on the trajectories of multiple vehicles ahead, or when there is no lead vehicle, we can utilize the trajectories of the vehicles following us. The discrepancy between the prediction results and the physics model is defined as a regularization term R_1:

$$R_1 = \sum\nolimits_{t\in\left[t_N^e, t_N^s\right]} \left(\widehat{v}_{Nt}^{\mathrm{Phy}} - \widehat{v}_{Nt}^{\mathrm{NN}}\right)^2 \# \tag{14}$$

The second part of physics prior is the kinematic constraint, ensuring that a vehicle's cumulative position over time aligns with its position at specific key moments. Therefore, during the training process of the NN, a well-predicted trajectory should avoid abrupt changes in speed or position at the endpoint in relation to the known trajectory. In other words, the error between the predicted endpoint and the actual endpoint position should be minimized:

$$R_2 = d_{t_N^s} - \left(d_{t_N^e} + \sum\nolimits_{t\in\left[t_N^e, t_N^s\right]} \widehat{v}_{Nt}^{\mathrm{NN}} \cdot \Delta \right) \# \tag{15}$$

With observed and collocation data defined, we are ready to define the loss function associated with the NN component and the physics-informed regularization. The loss function is a weighted sum of loss L and regularization terms as in Eq. (14) and (15). These two physics prior terms R_1 and R_2 could be regarded as two regularization terms. Unlike traditional regularization methods (e.g., L1 (Lasso) and L2 (Ridge) regularization), which penalize large weights in neural networks to reduce the complexity of the model and improve its generalization ability and extrapolation ability, R_1 and R_2 provide constraints on both the weights and biases of neural networks. Therefore, physical prior terms can be understood as a more effective form of regularization.

The weights for the physics prior terms in the PINN model is optimized by iterating over a range of potential values within the same scenario and dataset size, selecting the weights that yielded the best predictive performance of PINN.

The vehicle position predicted by the proposed PINN model is calculated based on the predicted velocities.

$$\widehat{d}_{Nt}^{\mathrm{PINN}} = d_{N\left(t_N^e-1\right)} + \sum\nolimits_{t'=t_N^e}^{t_N^s} \widehat{v}_{Nt'}^{\mathrm{PINN}} \cdot \Delta, \forall t \in \left[t_N^e, t_N^s\right] \# \tag{16}$$

4.2. Step 2 terminal state connection

In section 4.1, we combined physics priors with the NN model, which yielded trajectories with impressive predictive results. However, this approach still could not guarantee that the trajectories would align with their endpoints. Thus, we propose a kinematics-based criterion for connecting two broken trajectories through the stitched trajectories.

The kinematics-based criterion for trajectory connection is defined as follows. After we get the reconstructed trajectory continuing from trajectory K: $\left[\widehat{d}_{Nt}^{\mathrm{PINN}}\right]_{t\in\left[t_N^e, t_N^s\right]}$, the error between the reconstructed trajectory and surrounding trajectories $[d_{n't}]_{t\in\left[t_{n'}^b, t_{n'}^e\right]}$ is denoted as $\tau_{Nn'}$, which can be calculated by

$$\tau_{Nn'} := \sum\nolimits_{t=\max\left(t_N^e, t_{n'}^b\right)}^{\min\left(t_N^s, t_{n'}^e\right)} \left(\widehat{d}_{Nt}^{\mathrm{PINN}} - d_{n't}\right) \cdot \frac{\Delta}{\min\left(t_N^s, t_{n'}^e\right) - \max\left(t_N^e, t_{n'}^b\right)}, \forall N \in \mathscr{N}^b \# \tag{17}$$

If $\tau_{Nn'} < \epsilon^S$ and $\tau_{Nn'} = \min\{\tau_{Nn'}\}_{n'\in\mathscr{N}, n'\neq N}$, we consider the car-following characteristics of the two trajectories, trajectories N and n', are consistent. Thus, these two trajectories belong to the same vehicle and can thus be connected. ϵ^S is a given error term to evaluate the gap between the broken trajectory and the stitched trajectory. Selecting an appropriately sized ϵ^S is essential. A large value may cause incorrect connections, while a small value may reject correct connections. In the practice implementation, different ϵ^S values

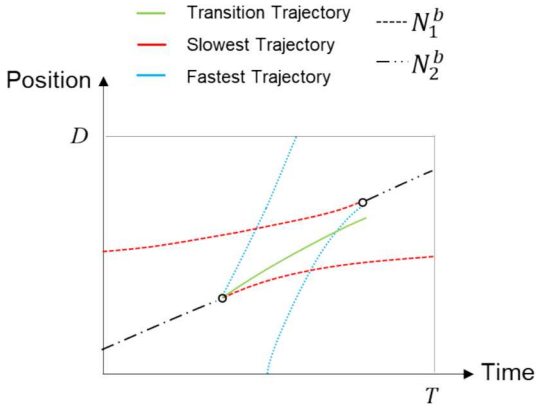

Fig. 7. Time-space cone illustration.

shall be given regarding the quality of the raw datasets. The ϵ^S values should be tailored based on the quality of the datasets in practical implementations.

It should be noted that stitched trajectory might not perfectly connect to the origin/end point of other broken trajectories due to estimation errors, as illustrated in Fig. 7. There is a gap between the stitched trajectory and the broken trajectory. To address this and connect the trajectory N and trajectory n', considering vehicle kinematics constraints. This method provides a structured approach to connect the trajectories in a manner that respects both temporal and spatial vehicle dynamics.

Assuming the kinematic constraints for each vehicle are provided, including the maximum speed v_{max}, maximum acceleration a_{max}, and deceleration a_{min}. For each pair of trajectories ready to be connected $i,i' \in \mathcal{N}^b$, we can uniquely generate two boundary trajectories. These trajectories begin at the end (or the origin) of the existing trajectories and are named the slowest and fastest trajectories, shown in Fig. 7. Each pair of the proposed boundary trajectories (a slowest trajectory and a fastest trajectory), which starts at the same point (e.g., either (t_{in_i}, y_{in_i}) or $(t_{i'1}, y_{i'1})$), covers all feasible trajectories passing the point, forming a cone-shaped region in the time–space graph.

The slowest trajectory is generated by moving the vehicle forward (or backward) with the maximum deceleration $\min\{a_{min}, 0\}$, which is 0 if the speed is zero) until the trajectory reaches time T (time 0) in the time–space graph. The slowest trajectory represents a lower bound for all feasible trajectories, indicating that any trajectory starting from the endpoint (or origin) operates faster than this trajectory. Conversely, the fastest trajectory is produced by moving the vehicle forward (or backward) with the maximum acceleration a_{max} until the vehicle reaches its maximum speed v_{max}. The fastest trajectory continues until it reaches time T (time 0) in the time–space graph. The physics meaning behind the fastest trajectory is that it serves as an upper bound to all feasible trajectories. This implies that any trajectory originating from the end (or the origin) of a given trajectory will operate at a speed slower than or equal to that of the fastest trajectory.

The equations (18)-(21) generates the slowest and fastest trajectories for sample i. To avoid redundancy, we only show the equations for sample N. The equations of trajectory n' can be derived similarly by considering the operation of the vehicle in reverse on the time–space graph.

$$d_{Nt}^{\text{slow}} = \begin{cases} d_{Nt} + v_{Nt}\delta - 0.5 \cdot a_{Nt} \cdot \Delta^2,\ t = t_N^e \\ d_{N(t-1)}^{\text{slow}} + v_{N(t-1)}^{\text{slow}}\delta - 0.5 \cdot a_{N(t-1)}^{\text{slow}}\delta^2, \forall t \in \{t_N^e + \delta, t_N^e + 2\delta, \cdots, t_N^s\} \end{cases} \# \tag{18}$$

$$v_{Nt}^{\text{slow}} = \begin{cases} \max(0, v_{Nt} - a_{min}\delta),\ t = t_N^e \\ \max\left(0, v_{N(t-1)}^{\text{slow}} - a_{min}\delta\right), \forall t \in \{t_N^e + \delta, t_N^e + 2\delta, \cdots, t_N^s\} \end{cases} \# \tag{19}$$

$$d_{Nt}^{\text{fast}} = \begin{cases} d_{Nt} + v_{Nt}\delta + 0.5 \cdot a_{Nt}\delta^2,\ t = t_N^e \\ y_{N(t-1)}^{\text{fast}} + v_{N(t-1)}^{\text{fast}}\delta + 0.5 \cdot a_{N(t-1)}^{\text{fast}}\delta^2, \forall t \in \{t_N^e + \delta, t_N^e + 2\delta, \cdots, t_N^s\} \end{cases} \# \tag{20}$$

$$v_{Nt}^{\text{fast}} = \begin{cases} \min(v_{max}, v_{Nt} + a_{max}\delta), if t \le \left\lceil \dfrac{v_{max} - v_{Nt}}{a_{max}\delta} \right\rceil \\ v_{max}, otherwise \end{cases} \# \tag{21}$$

As shown in Fig. 7, these boundary trajectories define a shadowed area within which all feasible trajectories must fall. These trajectories not only adhere to vehicle kinematics constraints but also effectively bridge the two broken trajectories. This area helps further avoids wrong connections.

By considering the stitched trajectory $\left[\widehat{d}_{Nt}^{\text{PINN}}\right]_{t\in\left[t_N^e, t_N^s\right]}$ that we obtained in Step 1, the old stitched trajectory, the new stitched trajectory that can connect the two broken trajectories (trajectories N and n') is obtained by the trajectory that has the minimum location difference from the old stitched trajectory in the shadow area. The new stitched trajectory is $\left[\widehat{d}_{Nt}\right]_{t\in\left[t_N^e, t_{n'}^b\right]}$. The equation to obtain the new stitched trajectory is shown in Equations (26) and (27). The speed of the new stitched trajectory can be obtained according to Equations (26) and (27) with a given location value.

$$\widehat{d}_{Nt} := \left\{ d'_{Nt} = \arg_{y_{Nt}^{\text{slow}} \le x \le y_{Nt}^{\text{fast}}} \min\left(\left|\widehat{d}_{Nt}^{\text{PINN}} - x\right|\right), \forall t \in \left[t_N^e, t_{n'}^b\right] \right\} \# \tag{22}$$

With this, the broken trajectories N and n' are connected. By applying a trajectory smoothing technique (Lee and Krumm, 2011; Li and Li, 2019) to the connected trajectory, the final trajectory is obtained, which is formed by combing the arrays of three trajectories, such as trajectories N, new stitched trajectory, and trajectory n'. By repeating these two steps, the issues we revealed previously, i.e., detection errors and limited ranges, can be successfully fixed. Specifically, due to the proposed connection criterion and the time–space cone, the misconnection situation can be overcome.

5. Experiment

5.1. Data preparation

In the numerical experiment, we demonstrate the proposed PINN vehicle trajectory stitching method with two sets of data. The first one is the widely-applied Next-Generation Simulation (NGSIM) dataset (NGSIM, 2007). To mitigate the influence of noises in the NGSIM data, we utilized a reconstructed version of the NGSIM US101 dataset that filters out noises (Dong et al., 2021). The second dataset is the HighSIM dataset (Shi et al., 2021). HighSIM dataset contains raw vehicle trajectory datasets extracted from aerial videos. The aerial videos were collected by three 8 K cameras on a helicopter from 4:15 – 6:15 pm on Tuesday, May 14, 2019, over 2,438-meters segment of Interstate-75 in Florida, United States. The vehicle trajectories operating from south to north are utilized in this paper. Both NGSIM and HighSIM used in this paper are 10 Hz data.

5.2. Test scenarios

We have designed five distinct scenarios to validate the performance of our proposed method for trajectory reconstruction in both unbiased (Scenario 0) and biased (Scenarios 1–4) sample conditions. In biased sample scenarios, the test samples feature rare conditions, such as high average acceleration (Scenario 1), low average acceleration (Scenario 2), high average speed (Scenario 3), low average speed (Scenario 4), while excluding these samples from the training set. This strategy is designed to evaluate the robustness of the proposed method under various sample biases. Each scenario is further introduced as follows:

Scenario 0. Unbiased: training, validation, and testing datasets are randomly selected. This serves as a baseline for comparison with other scenarios.

Scenario 1. High Acceleration Bias: the training dataset consists of samples with an average acceleration of less than 0.5, while the testing dataset includes samples with an average acceleration greater than 0.5. This scenario tests the method's ability to extrapolate from lower to higher acceleration regimes, as shown in Fig. 8 (a).

Scenario 2. Low Acceleration Bias: the training dataset is comprised of samples with an average acceleration greater than −0.5, and the testing dataset is composed of samples with an average acceleration less than −0.5. This tests the method's effectiveness in transitioning from mild deceleration to more pronounced deceleration, as shown in Fig. 8 (b).

Scenario 3. High-Speed Bias: the training dataset includes samples with an average speed of less than 11.5 m/s, while the testing dataset is characterized by an average speed greater than 11.5 m/s. This scenario evaluates the method's performance in extrapolating from lower to higher speed conditions, as shown in Fig. 8 (c).

Scenario 4. Low-Speed Bias: the training dataset includes samples with an average speed greater than 3.2 m/s and a testing

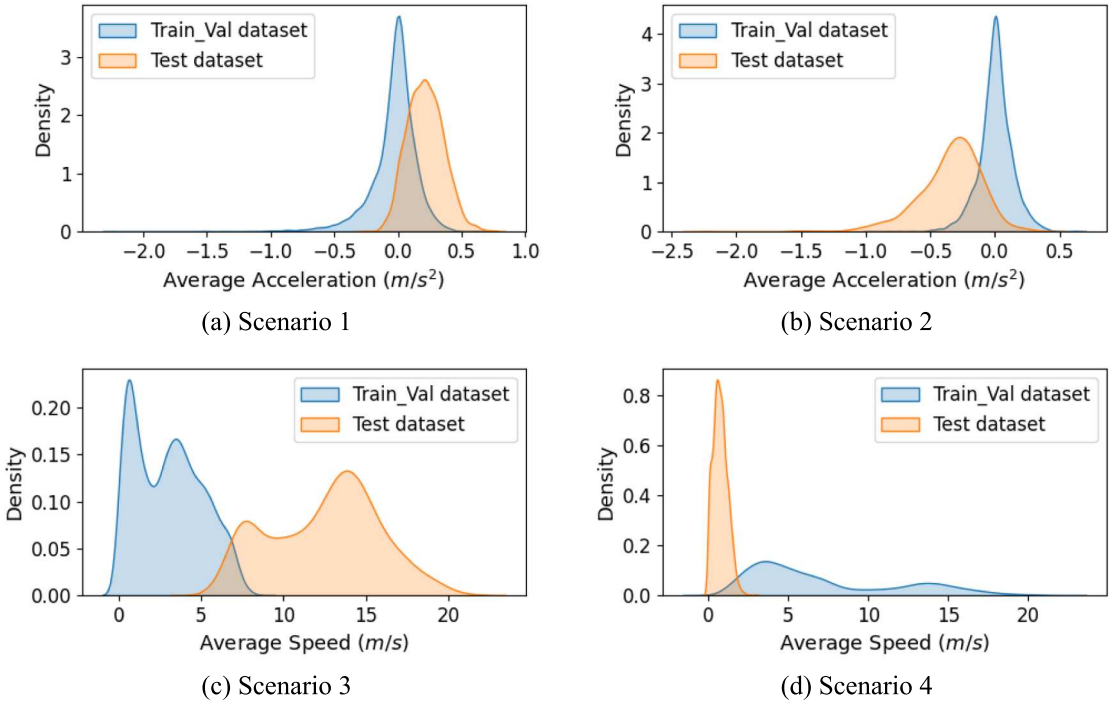

Fig. 8. Compare the train dataset and test dataset of four biased scenarios in NGSIM US101.

dataset with an average speed less than 3.2 m/s. It assesses the method's capability to adapt from relatively higher speeds to lower speed conditions, as shown in Fig. 8 (d).

5.3. Baseline models

To demonstrate the effectiveness of the proposed method, we compared it with both a **physics model** and **a pure NN model**. The physics model utilized the shockwave-based Newell's car-following model, augmented with the terminal state connection process in section 3.2. For the NN model, we chose a CLSTM model, which has the same NN component as the PINN model.

To compare the performance of two regularization methods, we also designed an ablation study, which aims to dissect the individual contributions of two physics regularization terms: R_1 and R_2 to the overall performance of their combined application. Thus, we test the performance of **PINN-R1**, which only contains the regularization term R_1 and **PINN-R2**, which only contains the regularization term R_2.

5.4. Model training settings

To ensure a fair comparison between the PINN model and baseline models (NN, PINN-R1, and PINN-R2), all these models were initialized randomly and did not undergo any pre-training phases. Throughout the training process, the NN and PINN models maintained identical architectural frameworks across all scenarios, and uniform training settings were applied. These measures were taken to ensure the integrity and fairness of the results comparison.

During the training process, we implemented a decreasing learning rate to prevent underfitting and employed early stopping to avoid overfitting. The specific parameters are outlined in Table 2: the initial learning rate was set at 0.001, it is reduced by a factor of 0.5 if there was no improvement in the model's performance on the validation set for 20 consecutive epochs. Early stopping is set to prevent overfitting by monitoring the model's validation loss and ceasing training if there is no improvement beyond a threshold of 0.00001 for 20 epochs. These strategies collectively ensure that each model, whether PINN-based or a baseline, is trained under optimal conditions to yield the best performance metrics, providing a fair and effective comparison of their capabilities.

For the hyperparameters, the weights of the physics prior terms in the PINN, PINN-R1, and PINN-R2 model, are chosen by iterating over a range of potential values within the same scenario and dataset size, selecting the weights that yielded the best predictive performance of PINN.

6. Experiment results

6.1. Trajectory stitching result

The proposed PINN method demonstrates superior prediction capabilities over two baseline methods (the physics method and the NN method) across all five scenarios when $t_N^2 - t_N^1 = 50; t_N^3 - t_N^2 = 20$, as shown in Table 2. In the unbiased scenario, the error rate of the PINN is 20 % lower than the Physics model and 10 % lower than the NN model in the NGSIM dataset. In the four biased scenarios, the PINN achieves much better predictions than the Physics model and NN model, and the two baseline methods show poorer predictions on the test set compared to the unbiased scenario. This superior performance of the PINN method is attributed to the physics model (shockwave-based Newell model) capturing the shock wave pattern across different dynamics scenarios. Therefore, it guides the model in making stable predictions in scenarios that exceed the training domain. Simultaneously, the physics constraints (kinematics constraints) act as a form of regularization, effectively avoiding overfitting and enhancing the model's extrapolation ability.

Regarding the prediction results for the HighSIM model, it is observed that HighSIM's overall prediction is slightly inferior to the results on the NGSIM data. A plausible explanation for this could be the degree of smoothness in vehicle speeds. Since the HighSIM data contains a higher proportion of high-speed trajectories (with vehicle speeds higher than 25 m/s accounting for 30 %), where vehicle trajectories tend to be smoother with fewer changes, predictions using learning-based methods are easier. In contrast, the NGSIM US101 data has lower vehicle speeds (around 12 m/s) with a greater proportion of trajectories featuring significant speed variations, resulting in less accurate predictions than HighSIM.

Table 3 displays the optimal weights of R1 and R2 obtained through tuning. Among the 5 test scenarios, both weights of PINN are greater than 0, indicating that each physics term makes a positive contribution to the training process within the loss function. Notice

Table 2
Hyperparameter Settings for Model Training.

Tuning parameter	Value
Batch size	256
Initial Learning rate	0.001
Learning rate decrease patience	20
Learning rate decrease factor	0.5
Early stop patient	20
Early stop delta	0.00001
Optimizer	Adam

Table 3
Prediction MSE of vehicle speed and position using NGSIM and HighSIM dataset.

Measure	Model	NGSIM					HighSIM				
		Scenario					Scenario				
		0	1	2	3	4	0	1	2	3	4
MSE of Speed (m^2/s^2)	Physics	0.208	0.319	0.559	0.632	0.425	0.172	0.286	0.492	0.532	0.366
	NN	0.156	0.24	0.419	0.474	0.319	0.137	0.212	0.359	0.424	0.262
	PINN-R1	0.139	0.234	0.425	0.410	0.263	0.124	0.209	0.358	0.366	0.223
	PINN-R2	0.137	0.234	0.430	0.394	0.246	0.119	0.210	0.362	0.359	0.207
	PINN	**0.132**	**0.225**	**0.411**	**0.368**	**0.228**	**0.115**	**0.198**	**0.352**	**0.333**	**0.190**
MSE of Position (m^2)	Physics	0.157	0.212	0.223	0.396	0.284	0.144	0.186	0.194	0.336	0.248
	NN	0.135	0.183	0.192	0.341	0.244	0.125	0.157	0.152	0.287	0.211
	PINN-R1	0.120	0.178	0.195	0.295	0.201	0.112	0.148	0.153	0.248	0.173
	PINN-R2	0.119	0.178	0.197	0.284	0.188	0.115	0.155	**0.155**	0.238	0.167
	PINN	**0.114**	**0.171**	**0.188**	**0.264**	**0.175**	**0.109**	**0.147**	**0.155**	**0.224**	**0.156**

that these weights represent proportions; since the two physical models (R1 and R2) embody different physical rule, comparing their magnitudes directly does not conclusively determine which physics rule is more critical.

These weights reflect the reliance of our method on different physical principles, demonstrating how physical constraints guide the model's learning process and help it to better understand and learn the underlying physical processes. When data is sufficient and overfitting is not a concern, these weights could approach zero, suggesting that the model does not require physical information to aid its training. Conversely, if we place higher trust in one type of physical information, its corresponding weight is increased to enhance the model's reliance on that specific physics insight. This approach balances data-driven insights with physical knowledge, optimizing model performance across various scenarios.

Fig. 9 shows the original velocity and position of a sample in the NGSIM data after steps 1 and 2, as well as the predicted values of the PINN model and the baseline model. From the speed prediction results, we can see that the physics model has the highest error, and the PINN model has the best prediction effect. All models capture the property of progressively larger velocities. The error in speed prediction is accumulated in the position, so the error is more obvious. In step 2, we process the prediction results in step one to ensure that the vehicle position coincides with the end position. This also inevitably sacrifices part of the continuity of the velocity. Among them, the physics model has a large error, causing the velocity to be modified a lot to match the final position. The more accurate the prediction effect, the smaller the magnitude of the modification. See (Table 4).

These results sufficiently prove that PINN can provide more accurate predictions for both speed and position, and it exhibits excellent predictive performance across a variety of situations that exceed the training domain. The results also highlight the ability of PINN to handle boundary conditions, complex-valued solutions, and various types of nonlinearities in governing ODE.

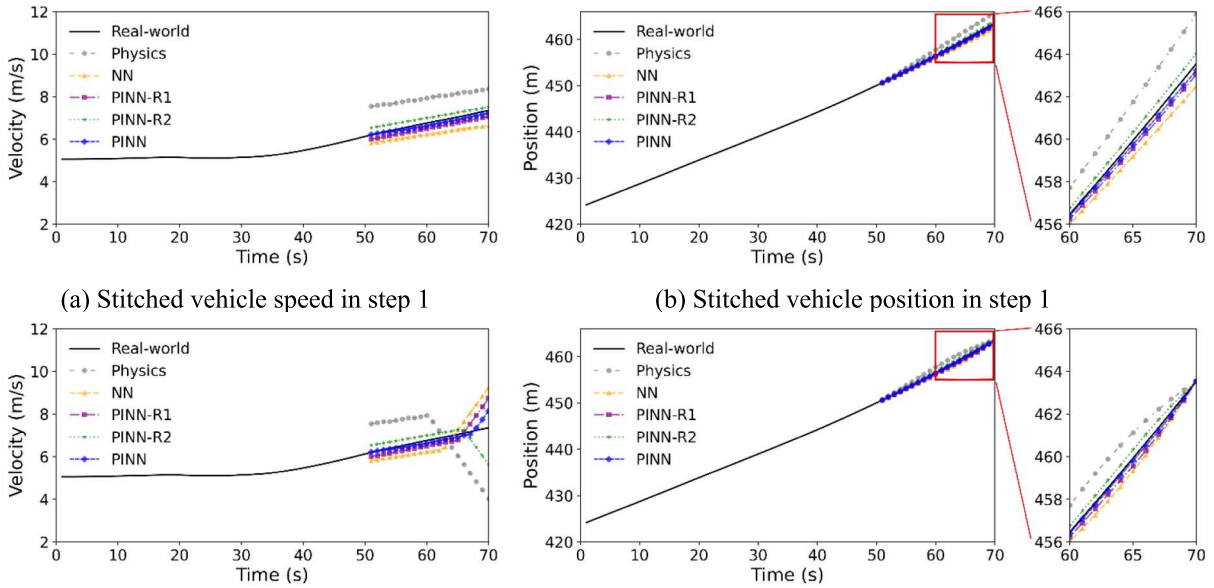

Fig. 9. Comparison of PINN results, baseline model results, and real-world data under Scenario 0.

Table 4
Tuned weights of the PINN model under different scenarios.

		Scenario				
		0	1	2	3	4
PINN-R1	w_1	0.04	0.005	0.015	0.01	0.025
	w_2	0	0	0	0	0
PINN-R2	w_1	0	0	0	0	0
	w_2	0.9	0.015	0.25	0.2	0.15
PINN	w_1	0.05	0.005	0.01	0.01	0.03
	w_2	0.3	0.05	0.15	0.2	0.1

6.2. Sensitivity analysis

6.2.1. Influence of training data size

This section evaluates the impact of training sample size on prediction performance, comparing five different training data sizes (600, 1200, 6000, 10000, 15000) using NGSIM data. The results are shown in Fig. 10. The results demonstrate that the prediction error decreases as the amount of data increases. The proposed PINN method consistently outperforms both the NN and the physics models. The performance of the physics model shows only a marginal increase in predictability with larger data sizes, stabilizing at a training data size of 5000. This is because the physics model used in this study is based on a single parameter (shockwave) and has a predefined reasonable range for this parameter, leading to effective training. In contrast, both the NN and PINN models show significant improvements in prediction performance as the training data size increases, with their performance stabilizing at a training data size of 5000.

6.2.2. Influence of broken time length

As the broken time length increases, the predictive performance of the PINN method decreases gradually, as shown in Fig. 11. However, even in different broken duration scenarios, PINN continues to outperform both the physics model and the NN model. With the increase in broken duration, the error of the physics model significantly grows, especially evident over a 5-second timespan, where the error accumulates. Meanwhile, both the NN and PINN models, leveraging their superior predictive capabilities, outperform the physics model when the broken time length is 50. Additionally, the incorporation of physics constraints in the PINN model enhances its stability, ensuring stable prediction results across various test scenarios.

6.2.3. Influence of the accuracy of the physics model

When incorporating physics models into neural networks, it is typically predicated on the assumption that the physics model possesses a certain level of accuracy to effectively aid the neural network. This underscores the critical importance of the accuracy of

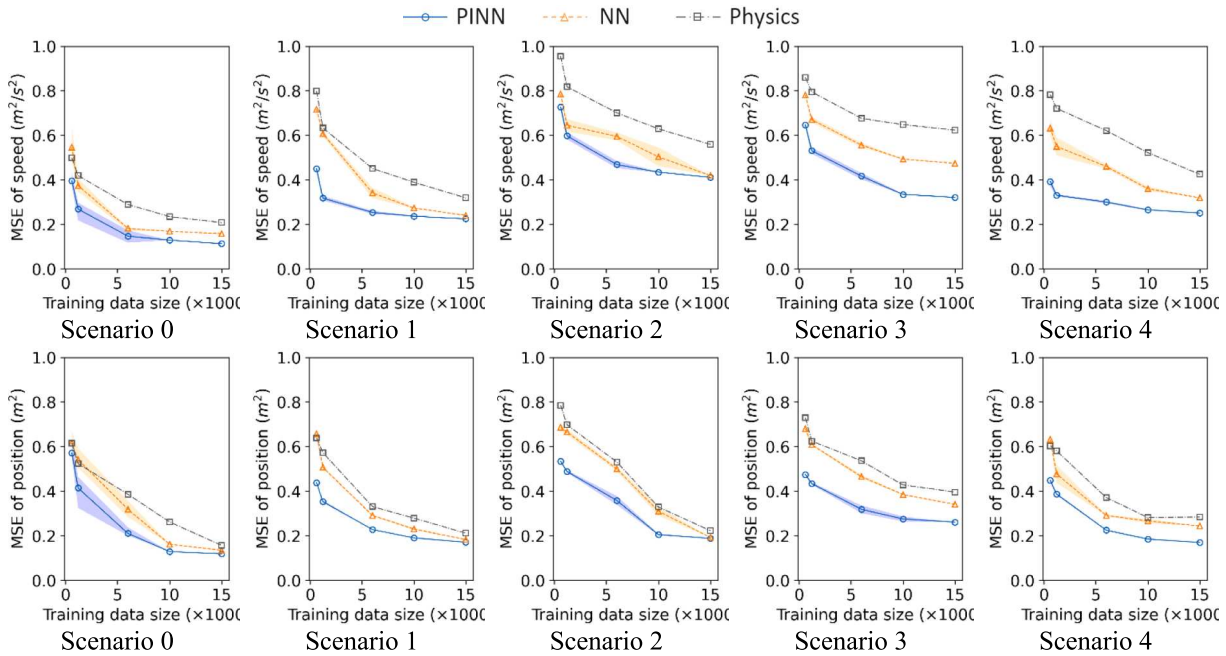

Fig. 10. The influence of training data size on prediction MSE of speed and position.

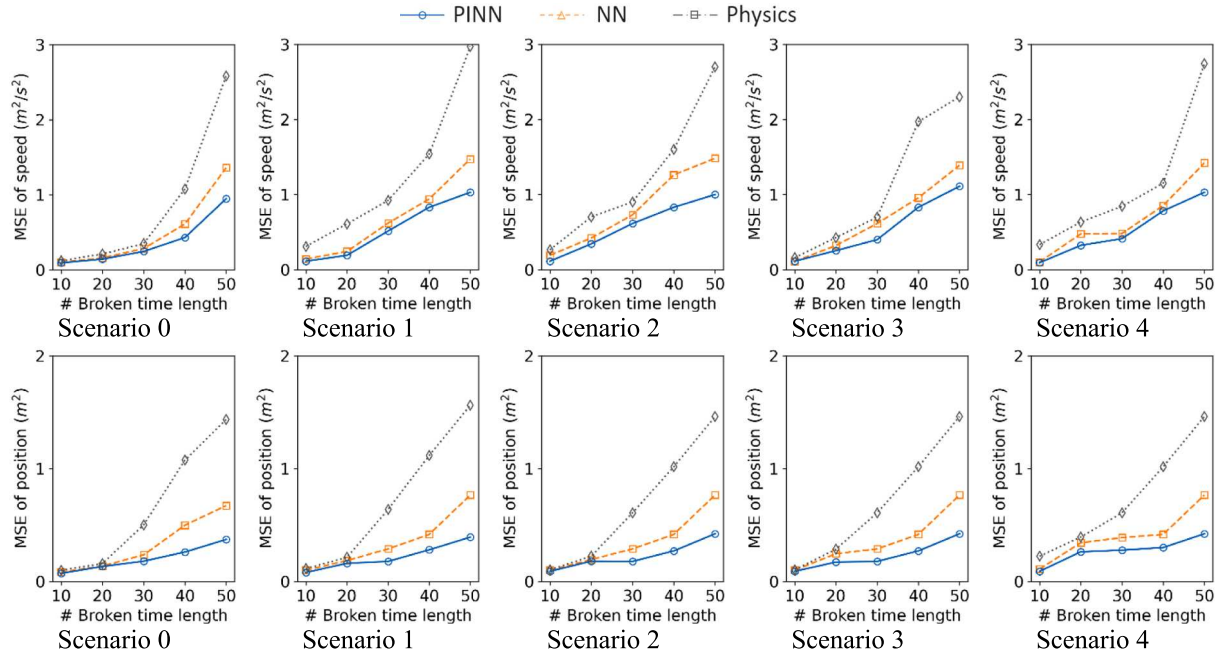

Fig. 11. The influence of broken time lengths on prediction MSE of speed and position.

the physics model. In our study, we tested the impact of varying accuracies of the same physics model (due to differences in parameters) on the performance of PINNs.

Results displayed in Table 5 indicate that the effectiveness of the PINN model is closely linked to the precision of the physics model. As the accuracy of the physics model improves, particularly when the shockwave speed parameter is adjusted from 4 m/s to 5 m/s, the speed Mean Squared Error (MSE) of the PINN model decreases to 0.124, demonstrating superior performance compared to the conventional Neural Network (NN) model. Conversely, when the error in the physics model is substantial, such as when the shockwave speed is 3.8 m/s, the contribution of the physics model becomes minimal and may even mislead the neural network model. Therefore, under such conditions, we reduce the weight of the physics component to zero. This adjustment effectively causes the PINN model to degenerate into a NN model. Thus, while the accuracy of the physics model is a critical factor, the PINN method safeguards the overall performance, ensuring that PINNs remain competitive with NN models even under less ideal conditions.

6.3. Processed trajectory data

Since we have access to the original video data from HighSIM, we further demonstrate the application of our proposed method in real-world scenarios using HighSIM. For details on the trajectory extraction process based on video, please refer to previous work (Shi et al., 2021).

Prior to the trajectory connection process, the raw dataset comprised 283,501 broken vehicle trajectories. These disruptions were primarily caused by issues such as missing detections and erroneous detections, as previously discussed. The speed and acceleration ranges of the trajectories in the raw datasets are [0, 150] ft/s ([0, 45.72] m/s) and [-20, 20] ft/s^2 ([-6.10, 6.10] m/s^2).

Using the proposed PINN trajectory connection method to process the extracted raw datasets, we processed the raw datasets, ultimately consolidating them into 2,184 complete vehicle trajectories. To illustrate the effectiveness of the proposed method, we included a visualization of the process vehicle trajectories in Fig. 12, compared to the unprocessed trajectories in Fig. 1.

In Fig. 1, the raw vehicle trajectories are fragmented into numerous small segments towards the end of the road section, a consequence of the camera angle issue described earlier in Fig. 1. Specifically, as vehicles move away from the camera, their apparent sizes on the video diminish, leading to decreased detection rates.

In contrast, Fig. 12 shows that after applying the proposed method, most vehicle trajectories are successfully connected, demonstrating the robustness and efficiency of our approach. A statistical analysis conducted by Shi et al. (Shi et al., 2021) further validates the quality of the HIGH-SIM dataset, indicating that it features more reasonable speed and acceleration distributions compared to the well-known NGSIM US-101 dataset. For additional details on this comparative analysis, interested readers are encouraged to consult the study by Shi et al., (2021).

7. Conclusion and future work

This paper proposed a PINN vehicle trajectory stitching method that can stitch broken vehicle trajectories using reconstructed

Table 5
Comparison of vehicle speed prediction of PINN, NN, and physics model with different shockwave speed w.

Shockwave speed w (m/s)	Physics	NN	PINN	w_1 in PINN	w_2 in PINN
3.8	0.228	0.156	0.156	0	0
4	0.214	0.156	0.141	0.05	0
4.2	0.211	0.156	0.135	0.05	0.05
4.4	0.208	0.156	0.124	0.05	0.05
4.6	0.207	0.156	0.132	0.05	0.2
4.8	0.208	0.156	0.129	0.05	0.2
5	0.207	0.156	0.128	0.05	0.1

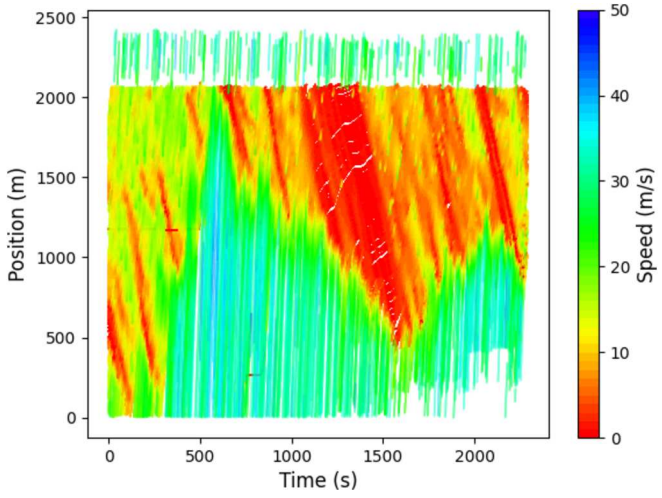

Fig. 12. Processed vehicle trajectory datasets.

trajectories. The PINN method enhances traditional neural networks by integrating physics priors, such as vehicle kinematics and boundary conditions. This approach aims to improve accuracy and extrapolation ability in cross-dynamics scenarios, such as extrapolating from low-speed training data to reconstruct high-speed trajectories. By incorporating these physics priors, the method extends beyond the training domain and provides effective regularization. To demonstrate the effectiveness of the proposed PINN method, it was applied to process a series of vehicle trajectory datasets derived from aerial videos captured over several successive spaces of Interstate-75, United States. When compared with various benchmark methods, the results indicated that the proposed PINN method excels in terms of both trajectory connection accuracy and consistency. The dataset processed using this method, the HIGH-SIM dataset, is recognized as the most extensive vehicle trajectory dataset to date, according to the authors' knowledge. It uniquely captures the full lifecycle of a traffic bottleneck. The dataset has been published online.

The broader application of PINN in other fields is influenced by the nature of the physical priors and data characteristics in the target field. For scenarios where physical priors are boundary conditions, we highly recommend employing PINN. Boundary conditions, as consistent rules, can provide a stable and reasonable regularization for the NN model, thereby enhancing the model's training effectiveness and stability. Furthermore, boundary conditions can also aid in data processing. When the physical priors are physical models, users might consider experimenting with PINN to see if the regularization provided by the physical model improves the NN model. When physical priors include both boundary conditions and physical models, detailed ablation studies, as done in this paper, are recommended to verify whether both types of physical prior are contributing to the model performance and generalization.

Future research can be conducted in several directions. Firstly, the physics-informed trajectory stitching model proposed in this paper is currently based on a single data source (video). Future work could consider applying the model to different data inputs and conducting data fusion. This is feasible as the NN component of the PINN model in this paper is capable of flexibly handling various types of data inputs. Secondly, while we have currently focused on predicting the micro-level characteristics of vehicle trajectory and already include both disaggregated and aggregated physics features, future research could extend to predicting or correcting macro-level traffic flow indicators. Thirdly, there are multiple ways to integrate physics priors into machine learning models, and the custom-designed loss functions and physics-based regularizations presented in this paper are among the most accessible and adaptable methods for various problems. Numerous other methods to incorporate physics into machine learning have been proposed in transportation and other fields (Kashinath et al., 2021). These include custom-designed neural network architectures (Wang et al., 2017) and the consideration of multi-scale properties. In the realm of traffic prediction, it is worthwhile to draw inspiration from sota approaches. By examining specific cases, we can explore the modeling performance and applicability of different Physics-Informed Machine Learning methodologies.

CRediT authorship contribution statement

Keke Long: Writing – original draft, Methodology, Conceptualization. **Xiaowei Shi:** Writing – review & editing, Validation, Supervision, Project administration. **Xiaopeng Li:** Writing – review & editing, Supervision.

Declaration of competing interest

This study was funded by National Science Foundation Cyber-Physical Systems (CPS) program, Award Number: 2343167, and Project 24UWM02 from the Center for Pedestrian and Bicyclist Safety (CPBS), supported by the U.S. Department of Transportation (USDOT) through the University Transportation Centers program.

References

Anuar, K., Cetin, M., 2017. Estimating freeway traffic volume using shockwaves and probe vehicle trajectory data. Transp. Res. Procedia 22, 183–192. https://doi.org/10.1016/j.trpro.2017.03.025.

Berghaus, M., Lamberty, S., Ehlers, J., Kalló, E., Oeser, M., 2024. Vehicle trajectory dataset from drone videos including off-ramp and congested traffic – analysis of data quality, traffic flow, and accident risk. Communications in Transportation Research 4, 100133. https://doi.org/10.1016/j.commtr.2024.100133.

Chen, X., Li, Z., Yang, Y., Qi, L., Ke, R., 2021. High-resolution vehicle trajectory extraction and denoising from aerial videos. IEEE Trans. Intell. Transp. Syst. 22, 3190–3202. https://doi.org/10.1109/TITS.2020.3003782.

Coifman, B., Wu, M., Redmill, K., Thornton, D.A., 2016. Collecting ambient vehicle trajectories from an instrumented probe vehicle: High quality data for microscopic traffic flow studies. Transportation Research Part C: Emerging Technologies 72, 254–271. https://doi.org/10.1016/j.trc.2016.09.001.

Dong, S., Zhou, Y., Chen, T., Li, S., Gao, Q., Ran, B., 2021. An integrated empirical mode decomposition and butterworth filter based vehicle trajectory reconstruction method. Physica A 583, 126295. https://doi.org/10.1016/j.physa.2021.126295.

Durrani, U., Lee, C., Maoh, H., 2016. Calibrating the Wiedemann's vehicle-following model using mixed vehicle-pair interactions. Transportation Research Part C: Emerging Technologies 67, 227–242. https://doi.org/10.1016/j.trc.2016.02.012.

Fang, S., Yang, L., Zhao, X., Wang, W., Xu, Z., Wu, G., Liu, Y., Qu, X., 2023. A dynamic transformation car-following model for the prediction of the traffic flow oscillation. IEEE Intell. Transp. Syst. Mag.

Hepburn, C.A., Montana, G., 2024. Model-based trajectory stitching for improved behavioural cloning and its applications. Mach Learn 113, 647–674. https://doi.org/10.1007/s10994-023-06392-z.

Jazayeri, A., Cai, H., Zheng, J.Y., Tuceryan, M., 2011. Vehicle detection and tracking in car video based on motion model. IEEE Trans. Intell. Transp. Syst. 583–595. https://doi.org/10.1109/TITS.2011.2113340.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L., 2021. Physics-informed machine learning. Nat. Rev. Phys. 3, 422–440. https://doi.org/10.1038/s42254-021-00314-5.

Kashinath, K., Mustafa, M., Albert, A., Wu, J., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., Manepalli, A., Chirila, D., Yu, R., Walters, R., White, B., Xiao, H., Tchelepi, H.A., Marcus, P., Anandkumar, A., Hassanzadeh, P., Lawrence, N., National, B., 2021. Physics-informed machine learning: case studies for weather and climate modelling Subject Areas.

Ke, R., Li, Z., Tang, J., Pan, Z., Wang, Y., 2019. Real-time traffic flow parameter estimation from UAV Video based on ensemble classifier and optical flow. IEEE Trans. Intell. Transp. Syst. https://doi.org/10.1109/TITS.2018.2797697.

Kim, Z.W., Cao, M., 2010. Evaluation of feature-based vehicle trajectory extraction algorithms. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC 99–104. https://doi.org/10.1109/ITSC.2010.5625278.

Kim, E.J., Park, H.C., Ham, S.W., Kho, S.Y., Kim, D.K., Hassan, Y., 2019. Extracting vehicle trajectories using unmanned aerial vehicles in congested traffic conditions. J. Adv. Transp. 2019. https://doi.org/10.1155/2019/9060797.

Lee, W.-C., Krumm, J., 2011. Trajectory Preprocessing, in: Computing with Spatial Trajectories. Springer New York, New York, NY, pp. 3–33. DOI: 10.1007/978-1-4614-1629-6_1.

Li, L., Li, X., 2019. Parsimonious trajectory design of connected automated traffic. Transp. Res. B Methodol. 119, 1–21.

Lim, Q., Lim, Y., Muhammad, H., Tan, D.W.M., Tan, U.-X., 2021. Forward collision warning system for motorcyclist using smart phone sensors based on time-to-collision and trajectory prediction. Journal of Intelligent and Connected Vehicles 4, 93–103. https://doi.org/10.1108/JICV-11-2020-0014.

Lin, H., He, Y., Liu, Y., Gao, K., Qu, X., 2023. Deep demand prediction: An enhanced conformer model with cold-start adaptation for origin–destination ride-hailing demand prediction. IEEE Intell. Transp. Syst. Mag.

Liu, Y., Lyu, C., Zhang, Y., Liu, Z., Yu, W., Qu, X., 2021. DeepTSP: Deep traffic state prediction model based on large-scale empirical data. Communications in Transportation Research 1, 100012. https://doi.org/10.1016/j.commtr.2021.100012.

Liu, Y., Wu, F., Liu, Z., Wang, K., Wang, F., Qu, X., 2023. Can language models be used for real-world urban-delivery route optimization? Innovation 4. https://doi.org/10.1016/j.xinn.2023.100520.

Long, K., Sheng, Z., Shi, H., Li, X., Chen, S., Ahn, S., 2024a. A Physics Enhanced Residual Learning (PERL) Framework for Vehicle Trajectory Prediction. DOI: 10.48550/arXiv.2309.15284.

Long, K., Shi, H., Chen, Z., Liang, Z., Li, X., de Souza, F., 2024b. Bi-scale car-following model calibration based on corridor-level trajectory. Transportation Research Part E: Logistics and Transportation Review 186, 103497. https://doi.org/10.1016/j.tre.2024.103497.

Mo, Z., Shi, R., Di, X., 2021. A physics-informed deep learning paradigm for car-following models. Transportation Research Part C: Emerging Technologies 130, 103240. https://doi.org/10.1016/j.trc.2021.103240.

Nabian, M.A., Meidani, H., 2020. Physics-driven regularization of deep neural networks for enhanced engineering design and analysis. J. Comput. Inf. Sci. Eng. 20, 1–10. https://doi.org/10.1115/1.4044507.

NGSIM, 2007. US Department of Transportation, NGSIM-Next generation simulation.

Punzo, V., Montanino, M., 2020. A two-level probabilistic approach for validation of stochastic traffic simulations: impact of drivers' heterogeneity models. Transportation Research Part C: Emerging Technologies 121, 102843. https://doi.org/10.1016/j.trc.2020.102843.

Punzo, V., 2009. Estimation of vehicle trajectories from observed discrete positions and Next-Generation Simulation Program (NGSIM) data.

Raissi, M., Karniadakis, G.E., 2018. Hidden physics models: Machine learning of nonlinear partial differential equations. J. Comput. Phys. 357, 125–141. https://doi.org/10.1016/j.jcp.2017.11.039.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045.

Raju, N., Arkatkar, S., Easa, S., Joshi, G., 2022. Developing extended trajectory database for heterogeneous traffic like NGSIM database. Transportation Letters 14, 555–564. https://doi.org/10.1080/19427867.2021.1908490.

Sazara, C., Nezafat, R.V., Cetin, M., 2017. Offline reconstruction of missing vehicle trajectory data from 3D LIDAR. IEEE Intelligent Vehicles Symposium, Proceedings 792–797. https://doi.org/10.1109/IVS.2017.7995813.

She, R., Ouyang, Y., 2024. Hybrid truck–drone delivery under aerial traffic congestion. Transp. Res. B Methodol. 185, 102970. https://doi.org/10.1016/j.trb.2024.102970.

Shi, X., Zhao, D., Yao, H., Li, X., Hale, D.K., Ghiasi, A., 2021. Video-based trajectory extraction with deep learning for High-Granularity Highway Simulation (HIGH-SIM). Communications in Transportation Research 1, 100014. https://doi.org/10.1016/j.commtr.2021.100014.

Tong, C., Chen, H., Xuan, Q., Yang, X., 2017. A framework for bus trajectory extraction and missing data recovery for data sampled from the internet. Sensors (Switzerland) 17. https://doi.org/10.3390/s17020342.

Victor, T., 2014. Analysis of Naturalistic Driving Study Data: Safer Glances, Driver Inattention, and Crash Risk, Analysis of Naturalistic Driving Study Data: Safer Glances, Driver Inattention, and Crash Risk. Transportation Research Board, Washington, D.C. DOI: 10.17226/22297.

Wang, J.X., Wu, J.L., Xiao, H., 2017. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. Phys. Rev. Fluids 2, 1–22. https://doi.org/10.1103/PhysRevFluids.2.034603.

Wang, G., Xiao, D., Gu, J., 2008. Review on vehicle detection based on video for traffic surveillance. In: In: 2008 IEEE International Conference on Automation and Logistics. IEEE, pp. 2961–2966. https://doi.org/10.1109/ICAL.2008.4636684.

Wang, G., Zhang, L., Xu, Z., Wang, R., Hina, S.M., Wei, T., Qu, X., Yang, R., 2023. Predictability of vehicle fuel consumption using LSTM: findings from field experiments. J. Transp. Eng., Part a: Systems 149, 04023030. https://doi.org/10.1061/JTEPBS.TEENG-7643.

Xiao, J., Xiao, Z., Wang, D., Havyarimana, V., Liu, C., Zou, C., Wu, D., 2022. Vehicle trajectory interpolation based on ensemble transfer regression. IEEE Trans. Intell. Transp. Syst. 23, 7680–7691. https://doi.org/10.1109/TITS.2021.3071761.

Xin, W., 2008. A Vehicle Trajectory Collection and Processing Methodology and Its Implementation to Crash Data 1–25.

Xu, Y., Yu, G., Wu, X., Wang, Y., Ma, Y., 2017. An enhanced viola-jones vehicle detection method from unmanned aerial vehicles imagery. IEEE Trans. Intell. Transp. Syst. 18, 1845–1856. https://doi.org/10.1109/TITS.2016.2617202.

Yao, H., Li, X., Yang, X., 2023. Physics-aware learning-based vehicle trajectory prediction of congested traffic in a connected vehicle environment. IEEE Trans. Veh. Technol. 72, 102–112. https://doi.org/10.1109/TVT.2022.3203906.

Zhang, G., Avery, R.P., Wang, Y., 2007. Video-based vehicle detection and classification system for real-time traffic data collection using uncalibrated video cameras. Transp. Res. Rec. 138–147. https://doi.org/10.3141/1993-19.

Zhang, T., Jin, P.J., 2019. A longitudinal scanline based vehicle trajectory reconstruction method for high-angle traffic video. Transportation Research Part C: Emerging Technologies 103, 104–128. https://doi.org/10.1016/j.trc.2019.03.015.

Zhang, J., Mao, S., Yang, L., Ma, W., Li, S., Gao, Z., 2024. Physics-informed deep learning for traffic state estimation based on the traffic flow model and computational graph method. Information Fusion 101, 101971. https://doi.org/10.1016/j.inffus.2023.101971.

Zhao, D., Li, X., 2019. Real-world trajectory extraction from aerial videos - a comprehensive and effective solution. 2019 IEEE Intelligent Transportation Systems Conference ITSC 2019, 2854–2859. https://doi.org/10.1109/ITSC.2019.8917175.

Zhao, H., Wang, C., Lin, Y., Guillemard, F., Geronimi, S., Aioun, F., 2017. On-road vehicle trajectory collection and scene-based lane change analysis: part I. IEEE Trans. Intell. Transp. Syst. 18, 192–205. https://doi.org/10.1109/TITS.2016.2571726.

Zheng, O., Abdel-Aty, M., Yue, L., Abdelraouf, A., Wang, Z., Mahmoud, N., 2023. CitySim: a drone-based vehicle trajectory dataset for safety-oriented research and digital twins. Transp. Res. Rec., 03611981231185768 https://doi.org/10.1177/03611981231185768.

Zhu, W., Wu, J., Fu, T., Wang, J., Zhang, J., Shangguan, Q., 2021. Dynamic prediction of traffic incident duration on urban expressways: a deep learning approach based on LSTM and MLP. Journal of Intelligent and Connected Vehicles 4, 80–91. https://doi.org/10.1108/JICV-03-2021-0004.