Vehicle Trajectory Interpolation Based on Ensemble Transfer Regression

Jianhua Xiao, Zhu Xiao, *Senior Member, IEEE*, Dong Wang, *Member, IEEE*, Vincent Havyarimana, Chenxi Liu, Chengming Zou, and Di Wu, *Member, IEEE*

***Abstract*—Vehicle trajectory collection usually faces challenges such as inaccurate and incomplete trajectory data, mainly due to missing trajectories caused by Global Navigation Satellite System (GNSS) outages. In this paper, a novel ensemble transfer regression framework is proposed for urban environments with transfer learning as the primary solution for constructing a fine-grained trajectory dataset during GNSS outages. First, GNSS and motion information are fused for the training process. Then, a regression-to-classification (R2C) process is employed to implement incremental training to adapt to dynamically changing environments. Third, to account for GNSS outages, transfer learning is integrated to construct a data filtering strategy that minimizes negative sample weights during the current scenario. Finally, a more accurate classification-type loss function for ensemble learning is designed to obtain the ensemble transfer regression model. We utilize real-world datasets to verify the accuracy of the comparative methods and the proposed framework in trajectory interpolation prediction. The experimental results show that our framework is significantly superior to the comparative methods.**

***Index Terms*—Vehicle trajectory collection, trajectory interpolation, transfer learning, ensemble learning.**

I. INTRODUCTION

A vehicle trajectory represents the changes in the vehicle position, speed, acceleration and acceleration rate over time [1]. In recent years, vehicle trajectory information has provided great potential and wide prospects in a variety of applications, including travel behaviour analysis, Internet of Vehicles, smart cities, and urban computing [2]–[5].

Developments in the above-mentioned applications have a common precondition: the trajectory collected from a vehicle must be complete and accurate. Nevertheless, in the widely used GNSS-based trajectory collection method, complex environments in urban areas, such as tall buildings and dense trees on both sides of the road, may hinder satellite signal reception, thereby leading to GNSS outages. As a result, the collected trajectory can be inaccurate or even lost [6], [7]. For example, a GNSS outage in an urban environment may result in a 60-second missing segment in the collected trajectory [8]. This increases the difficulty of extracting useful information from the trajectory. In addition, vehicle positioning errors inevitably occur in the presence of GNSS outages, and the deviation may be as high as 50 meters in some road sections of the collected trajectory data [9].

To obtain complete and accurate vehicle trajectory data, many trajectory interpolation techniques have been developed to tackle the issues of incompleteness and inaccuracy caused by GNSS outages in urban environments [10]–[13]. The most popular operation is to build a trajectory interpolation model based on machine learning by integrating vehicle positions (via GNSS) with motion information such as inertial measurements, including vehicle velocity and direction [14]. GNSS information and motion information are collected from different sources, i.e., different on-board devices such as GNSS receiver and vehicle motion sensors. The former obtains location coordinates, and the latter obtains information such as the speed, direction and acceleration of the vehicle. When GNSS outages take place in various road sections, the motion information retrieved via on-board devices [15] is able to provide supporting data that can be leveraged to correct the errors of vehicle positions via using the trajectory interpolation model.

Despite these efforts, the trajectory collection performance in urban environments is not guaranteed since the following challenges remain. i) The *retraining* problem [16]. When vehicles drive within an urban road network, road sections change rapidly, and such changes in the road conditions cause fluctuations in the motion information. To adapt to changes in road sections, existing methods need to implement retraining to gain the ability to update the trajectory interpolation model [17]. Nevertheless, during the retraining progress, the updated model may "forget" previously learned knowledge (in part or in whole). This inevitably degrades the trajectory interpolation performance [18]. ii) The *sample selection* problem [19]. The data samples provided for training can have a negative impact on the training results [20]. Most trajectory interpolation models use equal or random probabilities for sample selection. As a result, the model may learn from

Manuscript received XX XX, 2020.

This work was supported in part by the National Natural Science Foundation of China (Grants Nos. U20A20181, 61732017 and 61702175), in part by the Key Research and Development Project of Hunan Province of China (No. 2021GK2014), in part by the funding project of Zhejiang lab (No. 2021LC0AB05), in part by the Open Fund of State Key Laboratory of GeoInformation Engineering (No. SKLGIE2018-M-4-3), in part by the fund of the Hubei Key Laboratory of Transportation Internet of Things (No. WHUTIOT-2019004), and in part by the Natural Resources Scientific Research Project of the Department of the Natural Resources of Hunan Province (No. 201910). (*Jinahua Xiao and Zhu Xiao contributed equally to this work.*) (*The corresponding author of this paper is Zhu Xiao.*)

J. Xiao, Z. Xiao, D. Wang, C. Liu and D. Wu are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, 410082, China. J. Xiao is also with School of Computer Science and Engineering, Huaihua University, Huaihua, 418000, Hunan, China. e-mail: {xiaojianhua, zhxiao, wangd, cxliu, dwu}@hnu.edu.cn.

V. Havyarimana is with the Department of Applied Sciences, Ecole Normale Superieure, Bujumbura 6983, Burundi. e-mail: havincent12@gmail.com.

C. Zou is with Hubei Key Laboratory of Transport Internet of Things, School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China. e-mail: zoucm@whut.edu.cn.

Digital Object Identifier

some samples that actually deteriorate the performance of the training process [21]. Even though an ensemble learning framework is combined to set different weights for each sample, it is still challenging to select the proper samples [22].

To address the above-mentioned challenges, in this paper, we strive to design a novel vehicle trajectory interpolation framework enabling fine-grained vehicle trajectory collection in the meantime guaranteeing accuracy and completeness. The proposed framework mainly includes two modules, incremental training and data sample filtering. Firstly, we leverage the regression-to-classification (R2C) framework [23] to construct the retraining module. Specifically, we propose an ensemble transfer regression model by incorporating the R2C operation. By doing so, the proposed method has the ability to continuously remember the information in the fast changing road sections during the process of retraining. Secondly, We utilize the transfer learning, such as a common algorithm TrAdaboost [24], to construct the data sample filtering strategy. Via selecting the most relevant samples based data filtering, the proposed method is able to deal with the feature changes due to the varying of road conditions, thereby improving the interpolation accuracy for the inaccurate and incomplete trajectories. To summarise, we propose an ensemble transfer regression model, so as to solve the problems associated with trajectory interpolation during GNSS outages in urban environments. The main contributions of this paper are outlined as follows.

- We propose a vehicle trajectory interpolation framework based on ensemble learning to solve the retraining problem. The proposed framework implements incremental training by incorporating the regression-to-classification (R2C) framework to construct continuous data blocks (trajectory segments). Then, we traverse all the data blocks and evaluate the new data through existing submodels to learn new data blocks incrementally. Furthermore, incorrect and outdated submodels are discarded during the iteration process to enable the final ensemble model to capture changes in road sections.
- We design a data filtering strategy based on ensemble transfer learning with the purpose of resolving the sample selection problem for model training. The proposed strategy is to promote the model and focus as much as possible on positive samples by reducing the weights of negative training samples during the current training scenario. Moreover, the proposed method formulates different weight calculation rules for samples from different sources, which enhances the prediction accuracy.
- We conduct real-world trajectory interpolation experiments to evaluate the performance of the proposed ensemble transfer regression model. The experimental setup simulates the trajectory loss in an urban environment by enforcing random trajectory loss. The results demonstrate that the proposed method outperforms the existing methods and validates its excellent ability in applications to trajectory interpolation on various road sections covering fast lanes, low-speed lanes, turns and overpasses.

The remainder of this paper is organized as follows. Section II presents a brief review of related works. Section III introduces the proposed trajectory interpolation framework. Section IV presents experiments and discussions, and Section V concludes this paper.

II. RELATED WORKS

The development of emerging applications, such as location-based services (LBSs), smart cities and urban computing, is inextricably linked to vehicle trajectory data. However, due to the complexity of urban environments and unavoidable GNSS outages, the construction of accurate and complete trajectories is challenging.

Researchers have made efforts to address this challenge. One mainstream idea is to utilize the fusion of GNSS and inertial navigation system (INS) data to provide superior performance in comparison with either GNSS or INS standalone systems for trajectory interpolation. A fusion algorithm based on integrated GNSS/INS data is proposed in [25]; the algorithm is performed using a Kalman filter (KF) and requires a priori information related to the covariances of the data provided through both systems. In [26], the authors propose another GNSS/INS fusion strategy based on Kalman filter to solve the positioning enhancement when GNSS outages occur. To further improve the reliability and availability of GNSS/INS integration, an improved GNSS/INS integration scheme based on online support vector regression (OSVR) is proposed in [27], particularly for complex urban environments where GNSS signals are unavailable. In addition, motion information obtained through low-cost inertial measurement units (IMUs) and on-board diagnostics (OBD) of vehicles has also been widely used [28]. For example, ensemble learning method based on extended Kalman filter is introduced to evaluate the integration OBD speed measurements with missing GNSS data for vehicle trajectory estimation [29]. Another ensemble learning method such as Adaboost is developed in the OBD data applications [30]. In summary, as mentioned above, methods based on GNSS/INS fusion are good at effectively fusing data for trajectory interpolation. However, since these models will not distinguish and filter the fused motion information provided via INSs, IMUs or OBD, the performance of these models has not been significantly improved.

In order to better describe the potential relationship between historical trajectory and missing trajectory, many trajectory prediction methods have been proposed. The representative methods are neural network models based on the back-propagation neural network (BPNN) [31]. The drawback of the BPNN is that it needs a large amount of data for model training and network adjustment, and thus, the BPNN is prone to underlearning or overfitting. In [28], the authors regard the trajectory as a one-dimensional Gaussian process (GP), in which time is used as an independent variable; this method solves the nonlinear problem by traversing the entire trajectory to maximize the interpolation accuracy. In [32], a feed-forward artificial neural network method is proposed for navigating under complex urban environment. This method not only introduces the trajectory of a single vehicle, but also grasps the behavior of other vehicles in the driving environment and comprehensively predicts their trajectories. [33] designs an ensemble learning method based on support vector regression (SVR)

that forecasts the final destination by interpolating sparse taxi trajectories and solves nonlinear problems by mapping high-dimensional data onto two-dimensional planes. Another application uses the constructed artificial data set to obtain multi-scale position coding, and combines recurrent neural network (RNN) based on convolutional neural network to generate multiple reasonable future trajectories [34]. Alternatively, trajectory prediction can be transformed into sequence prediction. For example, a trajectory prediction method based on RNN is proposed in [35], which can learn prior knowledge from historical trajectory and transform it into target motion prediction. In summary, the above vehicle positioning methods are likely to be applied to trajectory interpolation in the future because of their good performance. However, these models do not consider the incremental training of the model and thus cannot adapt to a dynamically changing driving environment.

In addition to trajectory interpolation forecasting models, many studies have been conducted trajectory data mining, which could promote the practical applications with strict requirements on trajectory availability, such as autonomous driving. For example, in [36], two datasets are constructed to solve the 3D tracking and motion prediction for autonomous driving researches. Another example is that a high quality dataset based on camera images and radar data under diverse environments is proposed for tracking applications [37]. These methods devote a lot of effort to construct the availability of data set to ensure the automatic driving and its applications. Besides, there are many researches on the data processing framework. [38] introduces a trajectory preprocessing framework to analyze vehicle trajectory data that contains data cleaning and feature extraction. The authors [39] consider uncertain sensor data when mining vehicle trajectories and propose a dynamic programming-based algorithm to achieve the prediction of time-ordered location data. In [40], the authors focus on intersections and traffic rules and the trajectories generated by vehicles at turns and establish a linear reference system for automatically detecting roads. For nonlinear solutions, a regression framework integrated with topological learning is proposed in [41] that is able to realize ensemble topological learning neural networks and kernel density regression predictors. In summary, these methods have excellent trajectory data mining capabilities and provide the basis for trajectory interpolation. However, these approaches are based on the premise that the trajectory data studied are complete and accurate.

III. METHODOLOGY

In this section, we elaborate on the details of the proposed framework for trajectory interpolation.

Problem formulation. As illustrated in Fig. 1, the input includes GNSS data and motion information, namely, the trajectory point set $\{TP_i\}|_{i=0}^{N}$, vehicle velocity V, direction D and acceleration A ($\{(V_j, D_j, A_j)\}|_{j=0}^{M}$); The output is a series of trajectory points that can fill the missing part $\{TP_i\}|_{i=N+1}^{M}$. Let N and M denote the size of GNSS data and the motion information records, respectively, and $N < M$. After the GNSS outages take place, the size of the motion information records is $M - N$. When the data size N and M are determined, the output of the model is fixed as the trajectory of the predicted $M - N$ period as well. Then, the number of iterations t is determined, and a series of data are trained using the R2C framework to obtain multiple learners $(f_1, f_2, ..., f_t)$, thus realizing an incremental training process. Subsequently, in combination with TrAdaboost, we construct a data filtering strategy to filter out negative samples during the current training process and obtain the error rate set $\{\beta_1, \beta_2, ..., \beta_t\}$. Finally, we apply a weight voting mechanism to establish the trajectory interpolation model and incrementally interpolate the next trajectory and obtain a complete and accurate trajectory dataset.

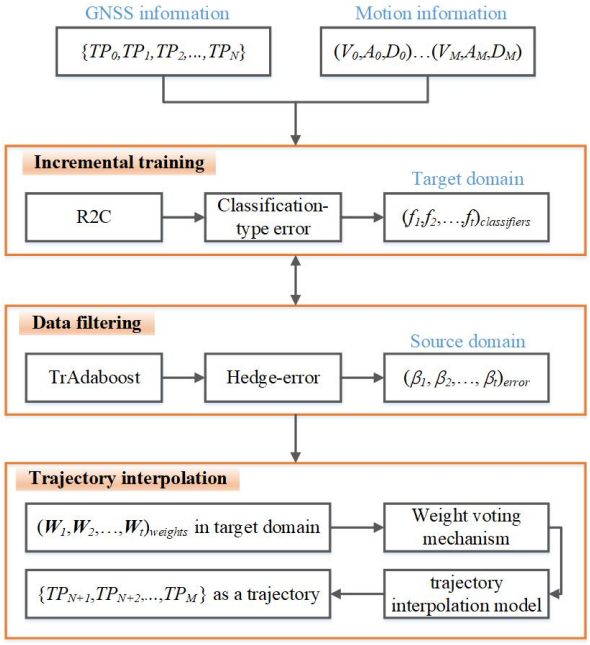


Fig. 1: Overview of the proposed vehicle trajectory interpolation model.

For the convenience of the following description, the parameters appearing in Fig. 1 and the possible parameters of each module included are defined. The notations of the parameters are given in Table I.

A. Incremental training process

The R2C framework is an adaptation of support vector machines (SVMs) [42], including linear and non-linear versions, to the Learn++ [43] framework. The SVM includes two versions for classification and regression, named support vector classification (SVC) and SVR, respectively, both of which are embodied in this framework. The R2C framework is unique because of the abilities to adjust the decision hyperplanes infinitely without retraining the complete model. For these reasons, it can be regarded as the perfect tool for the prediction of endless traffic trajectories. The detailed process of incremental training is as follows.

First, we divide the original regression input samples R into a series of input data blocks, i.e., $R = \{x^t(i) \in X; y^t(i) \in Y, i = 1, 2, ...n^t\}$, where n express the data block size. In each

TABLE I: Notations of the main parameters

Parameter	Definition
N	The size of the trajectory data
M	The size of the motion data
TP	The trajectory point set
t	The number of iterations (predictors)
n	The data block size for incremental training
X	The feature vector of motion information
Y	The distance vector of two consecutive trajectory point
X_d	The feature vector of the data different distribution
X_s	The feature vectors of the same data distribution
y	The response response variable for regression samples
y'	The class label for classification samples
$\hat{y}$	The class label for GNSS and motion information
P	The normal vector of SVC
b	The slope of the classification plane of SVC
ψ	A probability distribution of incremental training
ρ	A distribution of the current input data
φ	The error rate of incremental training
ϕ	The error rate of a testing
β	The weight adjustment factor of the X_s
ω	A sample weight in ensemble learning
$\mathbf{W}$	A learner's weight in ensemble learning
ϱ	A distance parameter for samples constructing
f	A trained learner for incremental training
F	A hypothesis (classifier) for the proposed method

iteration, we acquire the training datasets of the base learner by moving the samples up and down a small distance ϱ along the direction of the regression dimension, which is an adaptive parameter whose value is automatically and randomly chosen between 0 and 1. By assigning two values with different labels y', i.e., 1 and -1, the classification samples generated are shown below. Through allocating two values with different category labels, the generated classification samples C are expressed as follows.

$$C = \{([x^t(i), y^t(i) \pm \varrho], y'(i))^T \in \mathcal{R}^{n+1}, y'(i) \in \mathcal{R}\} \quad (1)$$

where $y'(i)$ represents the constructed classification label, and $y'(i) \in 1, -1$. After obtaining the classification samples, the base classifier is applied because a series of classification SVM learners with the current ϱ are trained on the complete dataset generated. The sample with $y_i + \varrho$ has the label +1, and the other classification generated by the same regression sample with $y_i - \varrho$ has the label -1. The separation plane and decision function of a single classification learner of the support vector classifier (SVC) are given by

$$P^T \mathbf{x} + b = 0 \quad (2)$$

where P is the normal vector of SVC and b is the slope of the classification plane of SVC. For each sample or data block, the estimated error rate φ for the prediction is calculated as follows.

$$\varphi = \sum_{i=1}^{k} \psi_i [f(\mathbf{x}_i) \neq y'(i)], \quad (3)$$

where ψ is a probability distribution, f indicates a trained learner and k is the size of the dataset.

Since the classification planes obtained by the ensemble are the same as the target regression planes, the classification is converted back to the regression to obtain the final ensemble regression function, which is constructed incrementally.

B. Data filtering process

When GNSS outage turns out, an attempt is made to generate an accurate trajectory interpolation result based on motion information, which requires motion information to replace GNSS data. For the model training, it needs to find positive samples in motion information that is consistent with the predictive distribution in GNSS, namely, to filter samples in short. Ensemble transfer learning can filter the samples with the sample weight updating mechanism. For example, reducing the weights of samples that have a negative impact on the training phase, thereby enabling the model to focus on the positive samples. Through screening out the negative samples and obtaining positive ones, the motion information is able to improve the accuracy of trajectory interpolation during GNSS outages.

In this subsection, we utilize the transfer learning framework based on Adaboost, namely, TrAdaboost [44], to construct a data filtering strategy through the targeted setting of sample weights. Two different types of training samples are involved in the proposed strategy. The first type comprises data that have the same distribution as the target field, called same-distribution training data. The same distribution data is used to assist the model training and transfer the motion information to the target missing trajectory segment. The second type consists of data that have different distributions from the target field, called diff-distribution training data. Diff-distribution data are used to determine the real classification target of the model. The characteristics of the data are marked in the training phase and unmarked in the test phase, corresponding to the data before and after the trajectory missing, respectively. Once the classification target of the model is determined, the same distribution data and different distribution data can be obtained. In the proposed method, Adaboost works on the same-distribution training data for constructing the base model. Then, a new weight mechanism based on $Hedge(\beta)$ [44] is developed for the diff-distribution training data so that the instances are the most dissimilar to the same-distribution data. A detailed description of the constructed strategy is presented in the following.

Suppose X_s is the same-distribution feature space, X_d denotes the diff-distribution feature space, and Y is the response variable. The final training datasets Υ_T are provided as follows.

$$\Upsilon_T \subseteq \{(X = X_s \cup X_d) \times Y\}, \quad (4)$$

where Y is a category label and can take $\{1, -1\}$ or others. In our work, X represents motion information such as (V, A, D), and Y represents distance difference calculated by latitude and longitude from the GNSS measurements.

Different weighted majority voting mechanisms are adopted for the two training datasets. On the one hand, for each iteration, the weight of the classifier on the same-distribution training data is computed by the classification error. The calculation error φ of an individual classifier F on the same-distribution training data is shown below.

$$\varphi^t = \sum_{i=n+1}^{n+m} \rho_i^t [F(X_i^t) \neq Y_i^t], \quad (5)$$

where ρ is a weight vector that satisfies a distribution ($\rho \in [0,1]$). On the other hand, for the diff-distribution training data, if the classifier is erroneously predicted in each iteration, its training weight is reduced by multiplying its weight by $\beta^{[F(X_i^t)\neq Y_i^t]}$ ($\beta^{[F(X_i^t)\neq Y_i^t]} \in (0,1]$).

C. Trajectory interpolation model

To address the problem caused by GNSS outages in trajectory interpolation, inspired by our previous work [23], we design a novel ensemble transfer regression framework, namely, a TrAdaboost and R2C structure, to realize incremental training and data filtering. Assume the explanatory variables X are N-dimensional input features, and $\{y\}$ is the corresponding response variable involved in the regression problem. The final target/output is a series of trajectory points to get the complete trajectory$\{TP\}$. In addition, there are two assumptions: X_s denotes the dataset with n-dimensional features from a source domain, and X_d denotes the dataset with m-dimensional features of a target domain. Algorithm 1 and Algorithm 2 present the trajectory prediction process for generating an accurate and complete vehicle trajectory, namely, the R2C-TrA method.

The design idea of the R2C-TrA method can be described in detail based on the following steps.

Step 1. The original regression samples $\xi^{\mathbf{R}}$ are divided into a series of training datasets $\xi^{\mathbf{R}} = \{(X_i, y_i)|X_i \in \mathcal{R}^{n+m}, y_i \in \mathcal{R}, i = 1,2,...,n+m\}$, where the first n data belong to X_s and the last m data come from X_d. At each iteration, the samples are moved up and down along the direction of the response variable's dimension over a small distance ϱ. Doing so provides accurate classification sample inputs for the framework. The classification samples $\xi^{\mathbf{C}}$ are transformed completely as follows.

$$\xi^{\mathbf{C}} = \{(\hat{X}^T, \hat{y}_i)|\hat{X}^T \in \mathcal{R}^{n+m+1}, \hat{y}_i \in \{+1,-1\}\}, \quad (6)$$

where i is the index of the samples and the maximum value is $2(n+m)$. The relationship between the original input and the new classification labels $\hat{y}$ is expressed as follows.

$$\hat{y}_i = \begin{cases} \hat{X}_i \in X_s \begin{cases} +1, & \hat{X}_i = [X_i^T, y_i + \varrho]; \\ -1, & \hat{X}_i = [X_i^T, y_i - \varrho]. \end{cases} \\ \hat{X}_i \in X_d \begin{cases} +1, & \hat{X}_i = [X_i^T, y_i + \varrho]; \\ -1, & \hat{X}_i = [X_i^T, y_i - \varrho]. \end{cases} \end{cases} \quad (7)$$

For the generated datasets, the classifier separation hyperplane by the new norm vector $\mathbf{P} = [P_X^T, p_y]^T$ is expressed by

$$P_X^T X + p_y y + b_{\mathbf{R}} = 0 \quad (8)$$

Step 2. The transfer learning algorithm, namely, TrAdaboost, is trained on the complete training datasets $\xi^{\mathbf{C}}$. Then, during the iteration process, the sample weights that are detrimental to the current scenario are reduced, and a series of SVCs are trained as base classifier learners. Finally, these classifiers are integrated by assigning appropriate weights ϕ whose values are obtained from the error calculations:

$$\phi_j = \sum_{i=n+1}^{n+m} \rho_i [F_j(\hat{X}_{d_i}) \neq \hat{y}_i], \quad (9)$$

Algorithm 1 R2C-TrA algorithm for transfer regression

Input: Two labelled datasets $\{X_s, y_s\}$ and $\{X_d, y_d\}$, the unlabelled datasets ξ. The maximum iteration value is T, and the block size is N_b. Weight adjustment factor of the source domain is β.

Output: Predicting a vector of trajectory points $\{y\}$.

1: Partition of the entire regression datasets with size N_b.
2: **while** Iteration **do**
3: **for** t=1,2,...,T **do**
4: Select ϱ, and for each $x \in (X_s \cup X_d)$ and $\hat{y} \in (y_s \cup y_d)$, construct the classification input as $x_i^t \in \{(x_i, \hat{y} \pm \varrho)|x \in \mathcal{R}^n\}$, $y \in \mathcal{R}$ and $y_i^t \in \{+1,-1\}, i = 1,...,(m_s + m_d)^t$. Assume $n = (m_d + m_d)$.
5: **if** $t = 1$ **then**
6: $\mathbf{P}^t = \rho^t(i) = \frac{1}{n^t}$ for $i = 1,2,...,n^t$, $\beta = \frac{1}{1+\sqrt{2\ln(n/T)}}$, then go to step 9.
7: **end if**
8: Apply the existing models F^{t-1} to evaluate $\{X_d, y_d\}$ on the new dataset and make error calculations.
9: $\phi^t = \sum_{i=1}^{n^t} \frac{1}{n^t}[F^{t-1}(x_i^t) \neq y_i^t]$, for $i = m_S + 1,...,n$.
10: Update and normalize the weights of instances. $\rho^t(i) = \frac{1}{n^t} \cdot \begin{cases} \beta, & \text{if } F^{t-1}(X_i^t) = y_i^t, 1 \leqslant i \leqslant m_S \\ \phi^t, & \text{if } F^{t-1}(X_i^t) = y_i^t, m_S + 1 \leqslant i \leqslant n \\ 1, & \text{otherwise.} \end{cases}$
11:
12: Update $\mathbf{P}^t = \rho^t / \sum_{t=1}^{n^t} \rho_i^t$ ($\mathbf{P}$ is a distribution).
13: **Call Algorithm 2** to train the classifiers and calculate the classifier weights W^t of the transfer framework.
14: Compute and obtain the final ensemble classification hypothesis.
15: $F^t(x_i^t) = \underset{y \in Y = \lceil T/2 \rceil}{\arg\max} \sum_{k=1}^{T} W_k^t I(f_k(x_i^t) = y)$
16: **end for**
17: **end while**
18: Obtain the complete trajectory vector $\{y\}$ with Eq. (12) and the trajectory point set $\{TP\}$.

where ρ is a probability distribution, $F(\hat{X})$ is the hypothesis that has been trained, and j is the number of existing classifiers ($j \leq (n+m)$). It is worth noting that evaluation predictions are performed only for each sample belonging to the target domain X_T. The sample weights ω_S from the source domain X_S are processed as follows.

$$\omega_S(i+1) = \omega_S(i) * \beta^{[F(\hat{X}_i)\neq \hat{y}_i]}, \quad (10)$$

where β is used to reduce the samples that are not related to the target domain, and its value is based on $Hedge(\beta)$.

Step 3. All available predictors are incrementally integrated through the ensemble learning weights $\mathbf{W}_T \propto 1/\phi$. The threshold value of predictor generated by each iteration is defined as 50%. This is because in binary classification, the average performance of a single weak predictor is with 50% accuracy. After determining the value of the threshold, perform the following operations: when the predictor's error rate is higher than 50%, the predictor will be discarded and

Algorithm 2 Transfer error calculation algorithm

Input: A base classifier f_k, the training sample $\{X_d, y_d\}$ and its size. The sigmoid parameters ϱ (slope) and a (infliction point). The conversion distance ϱ ($\varrho \in (0,1)$). The error boundary γ ($\gamma \in (0, 0.5)$).

Output: The trained classifiers and their weights.

Call BaseClassifier, providing it is combined with the current data and ξ is combined with the current ϱ to obtain $f^t : x \to \hat{y}$.
Evaluate all existing classifiers on $\{X_d, y_d\}$.
$\phi_k^t = \sum_{i=m_S+1}^{(m_S+m_T)^t} \mathbf{P}_i^t[f_k(x_i^t) \neq y_i^t]$.
if $\phi_{k=t}^t > 1/2$ **then**
    Discard f_k and go to Step 1 to generate a new f_k.
end if
if $\phi_{k<t}^t > 1/2$ **then**
    Set $\phi_{k<t}^t = 1/2$.
end if
Calculate the normalized error.
$\varphi_k^t = \phi_k^t/(1-\phi_k^t)$, for $k = 1, ..., t$, and set $\varphi_k^t \in [0, 1]$.
Calculate the weighted average of all normalized errors for the k^{th} classifier f_k : For $\alpha, a \in R$.
if $t = 1$ and $\varphi_{k=t}^t < \gamma$ **then**
    Set $\varphi_{k=t}^t = \gamma$, $W_{k=t}^t = \log(\frac{1}{\varphi_{k=t}^t})$.
end if
$\rho_k^t = 1/1 + e^{-\alpha(1-k-a)}$, $\rho_k^t = \rho_k^t / \sum_{j=1}^{t-k} \rho_k^{t-j}$.
$\bar{\varphi}_k^t = \sum_{j=1}^{t-k} \rho_k^{t-j} \varphi_k^{t-j}$, for k=1,...,t.
if $\bar{\varphi}_k^t < \gamma$ **then**
    Set $\bar{\varphi}_k^t = \gamma$.
end if
Compute and obtain the classifier voting weights.
$W_k^t = \log(1/\bar{\varphi}_k^t)$, for $k = 1, ..., t$.

reconstructed. After the iteration phase, the final regression plane (or a plane in the kernel space) is constructed, and the results can be obtained directly on the test data. The auxiliary domain (source domain) continuously adjusts the regression plane by reducing the sample weights after misclassification. In addition, according to Eq. (2), historical ensemble predictors are weighted in the model that best suits the current data sample, as described below.

$$\sum_z \mathbf{W}_d^z P_X^z X + \sum_z \mathbf{W}_d^z p_y^z y + b_{\mathbf{R}}^z = 0 \tag{11}$$

From Eq. (11), the regression response variable is hidden in the formula, and we can obtain the expression of y by applying a transformation, as shown in Eq. (12). This denotes the linear version of the regression function for integrating R2C and TrAdaboost.

$$y = \frac{\sum_z \mathbf{W}_d^z P_X^z X + b_{\mathbf{R}}^z}{-\sum_z \mathbf{W}_d^z p_y{}^z} \tag{12}$$

The above framework can be applied only to linear regression models. In contrast, a kernel function is employed to construct a regression model for the nonlinear case. Unfortunately, the nonlinear kernel functions of traditional SVMs, such as the polynomial kernel, RBF kernel, and sigmoid kernel, cannot be directly applied to the proposed framework. To resolve this issue, we seek to solve an equation and then obtain the corresponding response variables in the kernel space in the R2C and TrAdaboost framework. When the kernel function is introduced, the ensemble separation planes based on the classification task are obtained as follows.

$$\sum_z \mathbf{W}_d^z (C^T)^z \mathbf{K}(\hat{X}_V^z, \hat{X}) + b_{\mathbf{R}} = 0, \tag{13}$$

where $\mathbf{K}$ is the kernel function, C is the penalty factor representing the degree of punishment for the classification error in the case of linear inseparability, and $\hat{X}_V$ presents the support vectors obtained through SVM training [42].

Taking the polynomial kernel function $K(X_V, X) = (\gamma X_V^T X + r)^q$ as an example, Eq. (13) can be expressed as follows.

$$\sum_z \mathbf{W}_d^z (C^T)^z (\gamma (\hat{X}_V^z)^T \hat{X} + r)^q + b_{\mathbf{R}} = 0 \tag{14}$$

Note that the response variables are hidden in the support vectors. We separate the explanatory and response variables and make the following definitions. First, $\hat{X}(i)$ is represented as the ith component of the array $\hat{X}$. Then, $\hat{X}_V(i,j)$ is expressed as the jth attribute value of the ith support vector. Third, the length of definition C is k. In the end, we obtain the equations shown below.

$$\sum_z \mathbf{W}_d^z (C^T)^z \begin{pmatrix} (\sum_j \hat{X}_V^z(1,j)\hat{X}(j) + r)^q \\ \cdots \\ (\sum_j \hat{X}_V^z(m,j)\hat{X}(j) + r)^q \\ (\sum_j \hat{X}_V^z(m+1,j)y + r)^q \end{pmatrix} + b_{\mathbf{R}} = 0 \tag{15}$$

Through deriving Eq. (15), the final estimate y can be obtained by

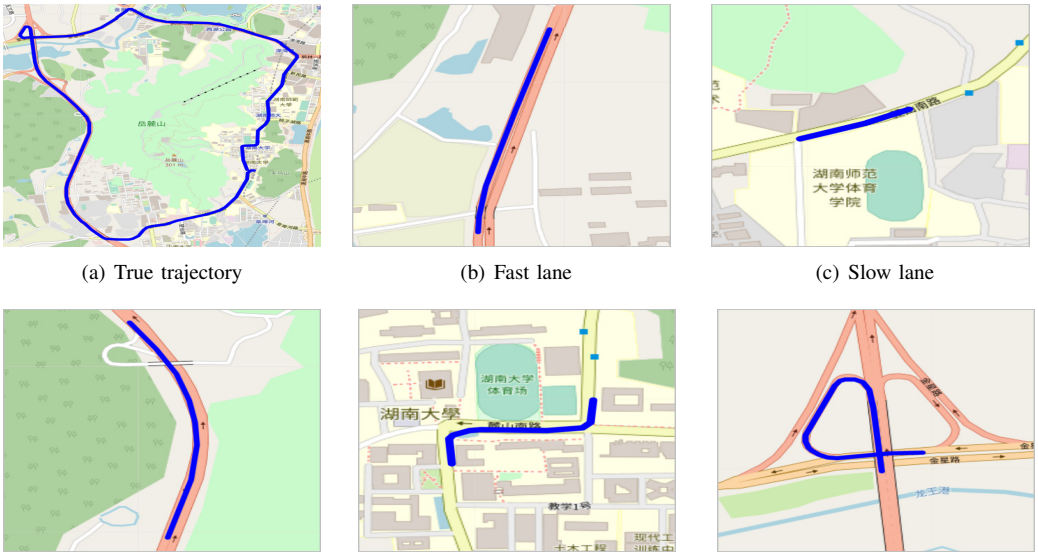
$$y = \frac{\sqrt[q]{\frac{-b_{\mathbf{R}}}{\sum_z \mathbf{W}_d^z}} - (\sum_t C_t^k \sum_j^{k-1} \hat{X}_V^z(t,j)\hat{X}(j) + r)^q) - r}{C_k^z \sum_j \hat{X}_V^z(m+1,j)} \tag{16}$$

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we conduct real-world experiments to validate our trajectory interpolation prediction model. After acquiring the vehicle trajectories, we artificially create GNSS signal interruption to simulate the loss of trajectory under different urban road scenarios. The parameter adjustment and optimization are given by employing the Bayesian optimization method. After the parameters are selected, the trajectory interpolation comparison results are given, which are reflected by an evaluation metric. In the paper, we transform a series of continuous trajectory points into the distance difference, which is used as the response variable of the model for regression prediction. Therefore, the comparison measure is chosen as the root mean square error (RMSE).

$$RMSE(X, H) = \sqrt{\frac{1}{N}(\sum^N (H(X) - Y)^2} \tag{17}$$

where H is a predictive model, X expresses the dataset of motion information $(\{(V_j, D_j, A_j)\}|_{j=0}^M)$, N displays the size of X, and Y indicates a real trajectory point set.



(a) True trajectory (b) Fast lane (c) Slow lane

(d) Turning (e) Continuous turning (f) Overpass

Fig. 2: Road sections for the trajectory interpolation experiments.

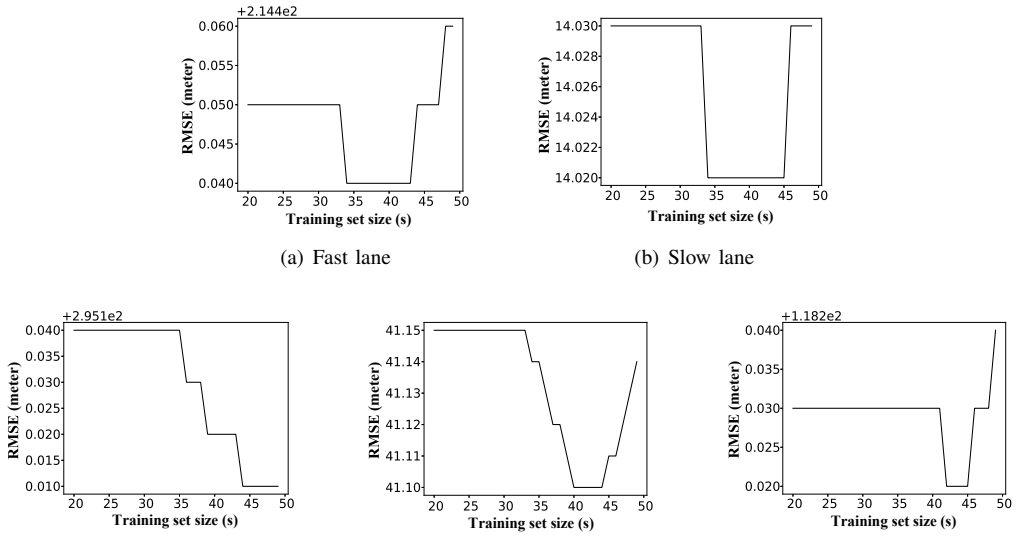
A. Experimental setup

We evaluate the proposed framework by performing experiments and comparative studies. GNSS and motion information involved in experiments are gathered from real-world vehicles. In our road test, the motion information of the vehicle is obtained through the OBD reader, which is low-cost auxiliary information acquisition equipment. The motion information records, including velocity, direction and acceleration ($\{(V_j, D_j, A_j)\}|_{j=0}^{M}$), offer useful information needed for model training when the GNSS outages occur. Note that trajectory data will be with large position error or even lost during GNSS outages, which brings the inaccuracy incompleteness of the trajectory data. Depending on the motion information mentioned above, the position deviation in the future can be predicted. For each selected scenario, 70% of the complete trajectory is the training set, the rest is the validation set, and the test set depends on the time of GNSS outage. Therefore, the prediction target of this study is defined as the offset distance between two continuous trajectory points. This requires that the original trajectory data should be processed as the distance between two consecutive points, so as to determine the prediction target Y (show as Table I) of the comparative experiment.

The experiments are roughly divided into the following steps. First, a specific road section is selected (e.g., a straight lane, an overpass, a fast lane, a slow lane, a turn, or a continuous curve). In the selected road section, it can be divided into two types of scenarios. One is the simple road section with little change in driving direction, such as high speed, low speed and curve. These road sections are used to verify the incremental training ability of the proposed method. The other is the road section with complex driving environment and changeable driving direction, such as continuous turning and overpass, which are used to test the performance of the data filtering module of the proposed method. Second, the data before GNSS outage is used as training data, and then the OBD data is fused to train the model. Finally, when entering the GNSS interruption period, during which the GNSS outages occur, the missing trajectories are obtained through OBD data, namely the motion information.

The trajectories are collected by low-cost GNSS receiver and OBD device in Changsha city, China. In total, we present five road sections, as shown in Figs. 2(a). To specify, straight line sections contain fast lanes where the speed changes smoothly and slow lanes where the speed is relatively low. The application scenarios on curved roads include turning, continuous turning and overpasses. The five road sections are illustrated in Fig. 2, where the data loss on Fig. 2(b) is set as 38 seconds, the loss on Fig. 2(c) is set as 40 seconds, the losses on Fig. 2(d) and Fig. 2(e) scenarios are 44 and 58 seconds, respectively, and the loss on Fig. 2(f) is 68 seconds.

For the comparative study, several mainstream approaches are selected. i) Conventional regression methods such as Gaussian process regression (GPR) [28]; this method is the most commonly used machine learning method for regression prediction. ii) Artificial neural network methods, such as back-propagation neural networks (BPNNs) [31], which train a multilayered neural network to adapt to internal representations and learn any mapping from input to output. Another neural network method based on sequence prediction, iii) the recurrent neural network (RNN) [35] with three layers and 30 states is selected. In addition, two ensemble learning methods, including iv) extra tree regression (ETR) [45] and v) SVR-based Adaboost [30], and two incremental learning methods, vi) OSVR [27] and vii) PAR [46], are employed in our experiments. These methods highlight the advantages of ensemble learning and incremental learning and ensure the adequacy and fairness of the experiments. In our experiments, We first select a data block before GNSS outage to train the model. The trajectory in Fig. 2(b)-(f) are the road section with



(a) Fast lane (b) Slow lane

(c) Turning (d) Continuously curving (e) Overpass

Fig. 3: RMSEs with different window size settings.

GNSS outages, which will not be used as training data for model training. In our proposed method, the GNSS data (TP) and the OBD data (V, D, A) before the GNSS outages occur, are fused by using transfer learning. The prediction model is obtained, which can interpolate the missing trajectory of Figs. 2(b)-(f) with OBD data. Then the relative parameters are tuned with Bayesian parameter optimizations. Finally, all the trained models are interpolated on the missing trajectories on various road sections to obtain experimental comparison results in terms of evaluation metrics.

B. Parameter setting

In the proposed method, the parameters that need to be adjusted are the window size N_b (block size), the penalty factor C in SVMs, and the parameters in different kernel functions. For example, the polynomial kernel has three parameters: γ, the degree d and r.

The selection of the window size N_b can be carried out through cross-validation, where the error rate is set to the root mean square error (RMSE) obtained on the dataset. The experimental results in five application scenarios are shown in Fig. 3. Note that the training dataset for this experiment comprises the data before the missing trajectory. Therefore, each mark on the abscissa indicates that the amount of training data increases by one second.

In addition to the window size N_b, the parameters involved in our experiments are determined through Bayesian optimizations [47] and are used for the interpolation prediction experiments of the model on missing trajectories. The main reason is that these parameters have little effect on the overall model and are difficult to determine through cross-validation. Therefore, Bayesian optimization is incorporated in the proposed framework to determine these parameters.

As seen from Fig. 3, the window sizes are verified in various road sections, and the overall trends are similar. Although a larger window size will improve the prediction accuracy, when the value exceeds a certain critical point, it will decrease the forward transfer ability of the transfer learning algorithm. In the experimental results, the critical point appears as 43 seconds based on the experiments described in the next subsection.

C. Results and discussion

The experimental results are illustrated in Figs. 4-8. In these results, the black trajectory represents the real acquired trajectory, which serves as the reference. Our methods are linear and nonlinear R2C-TrA, which correspond to the red trajectory and the purple trajectory, respectively. Two incremental learning methods, PAR and OSVR, are expressed as the cyan and yellow trajectories, respectively. Two ensemble learning methods, Adaboost and ETR, are displayed as blue and brown, respectively. The results of RNN method are denoted with brick red. With regard to the two conventional regression methods, the GPR method is indicated by the orange trajectory, and the BPNN is shown as the green trajectory.

Fig. 4 presents the trajectory interpolation results on the fast lane. The original trajectory is a city road limited to a speed of 70 km/h, which defines a high-speed road in an urban environment. From the results, except for the slight bias in the trajectories with respect to the GPR and BPNN methods, the trajectories obtained by the other methods are highly similar to the real trajectory, especially the trajectories of the proposed methods, which almost overlap with the real one. The main reason is that the changes in vehicle velocity are more stationary, and the time offset is negligible.

Fig. 5 presents the results obtained from the slow lane. There are sidewalks on either side of the road section; hence, the velocity is slower, even approaching zero. Based on the results in Fig. 5, the trajectories acquired by GPR, Adaboost, RNN and ETR show the most obvious fluctuations because

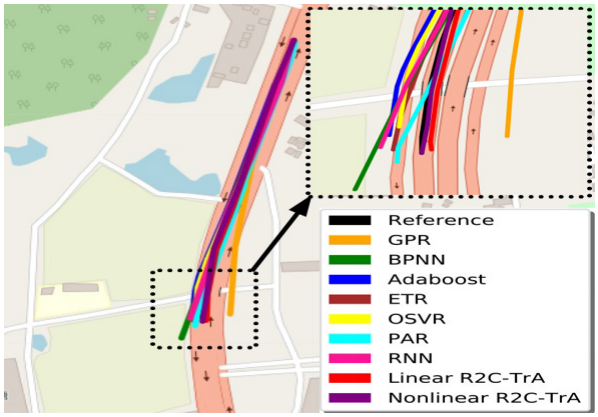


Fig. 4: Trajectory interpolation results on the fast lane.

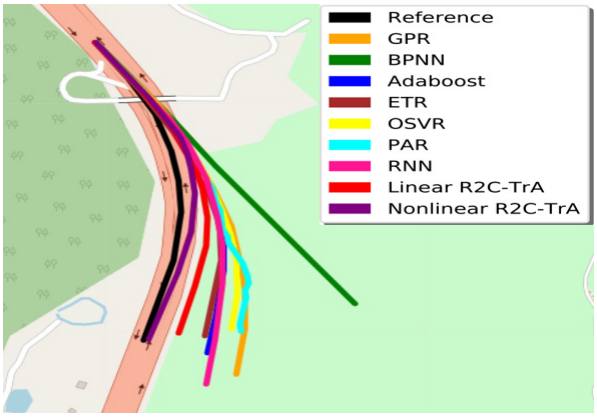


Fig. 6: Trajectory interpolation results on the turning road.

the trajectory points are dense and the vehicle velocity is often slow on the whole road. Especially for the prediction results of BPNN and RNN, there are obvious shakes at every speed of zero. This is because the method based on neural network is tolerant of outliers. A fact presented for the reference trajectory is that for the sidewalk near the right, the speed is frequently 0. Therefore, the results shown in the Fig. 2(c), is that when the real trajectory is close to the right-hand sidewalk, the trajectories produced by PAR, OSVR and BPNN have an obvious deviation. The best effect of OVSR is due to the role of incremental training, which is also reflected in the proposed methods linear R2C-TrA and nonlinear R2C-TrA. Apparently, nonlinear R2C-TrA achieves the most accurate results.

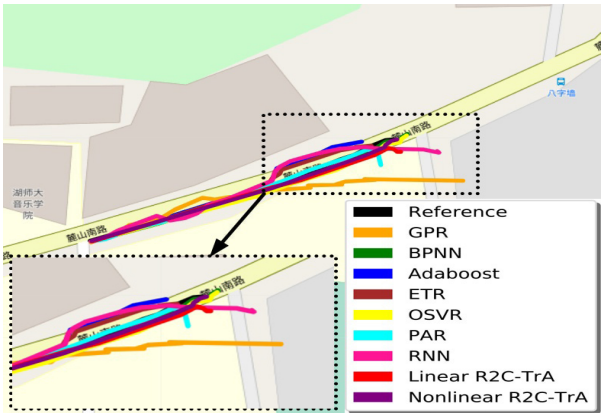


Fig. 5: Trajectory interpolation results on the slow lane.

Fig. 6 illustrates comparison results on the turning road. When the test vehicle drives through this turning section, the vehicle velocity is controlled within 40 km/h. From Fig. 6, it can be seen that the trajectory obtained by nonlinear R2C-TrA is the closest to the reference, followed by that obtained by linear R2C-TrA. In addition, the results of the ensemble learning methods are slightly superior to the incremental learning results, while the conventional regression methods are inferior to the other methods because the speed and direction of the vehicle are constantly changing along this long turn, and the ensemble learning methods are better at adapting to these changes. Because of the recurrent network structure, the RNN can achieve better results than incremental learning methods, but slightly worse than ensemble learning methods. This is because the recurrent network structure are vulnerable to environmental changes. Furthermore, the proposed methods are superior to ensemble learning because the transfer process in our framework minimizes the sample weights of training samples that would be harmful to the current scenario.

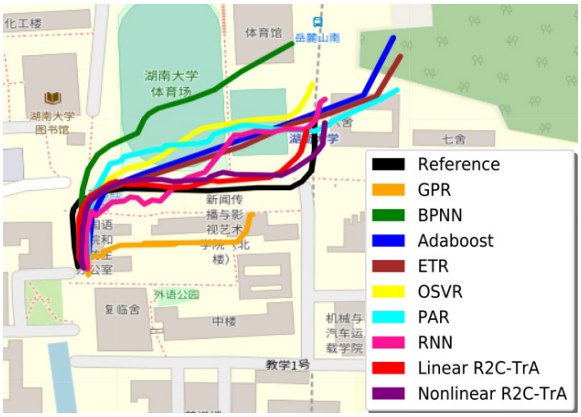


Fig. 7: Trajectory interpolation results on the continuously curving road.

Fig. 7 presents the interpolation results in a more complicated situation, that is, the continuously curving road. From the results, GPR, OSVR and RNN have similar contours to the reference trajectory, although they strongly deviate up and down. Although RNN jitters severely, its prediction performance in this scenario is the best in the baselines due to its ability of sequence prediction. Furthermore, the two ensemble learning methods have similar trajectories, while they show great deviations. The two trajectories obtained by BPNN and PAR are the most dissimilar with the reference trajectory. In addition, the two proposed methods are the closest to the real trajectory; the main reason behind this is that during the training process, transfer learning is able to eliminate the most negative samples, making the model more adapt to the diverse changes in road sections. In summary, the proposed framework incorporates transfer learning and thus is more applicable to different road sections in vehicle trajectory interpolation

TABLE II: The trajectory interpolation errors with the RMSE

Methods	**RMSE (meter)**					
	Fast-lane	Slow-lane	Turning	Continuously curving	Overpass	**Mean**
GPR	26.04	16.20	69.26	79.32	70.22	52.21
BPNN	22.21	6.19	125.70	81.73	117.76	70.72
Adaboost	16.32	16.35	50.07	43.11	46.70	34.51
ETR	13.75	14.87	46.65	42.59	50.26	33.62
OSVR	20.04	6.81	63.38	45.87	81.80	43.58
PAR	6.05	9.00	65.35	47.63	66.00	38.81
RNN	5.53	12.57	55.36	38.12	41.02	30.52
Linear R2C-TrA	5.29	4.01	33.90	27.24	28.19	19.73
Non-linear R2C-TrA	3.92	1.57	13.84	20.07	23.66	12.61

applications and demonstrates excellent advantages.

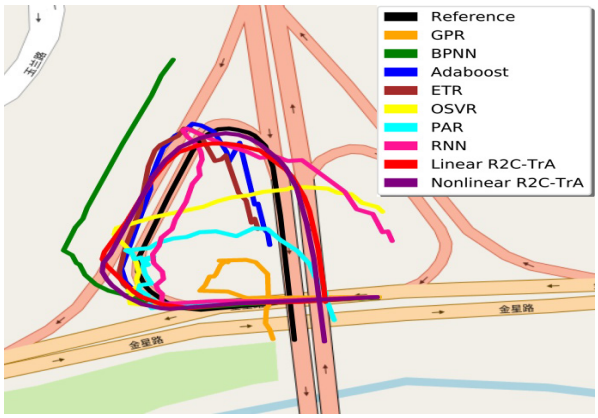


Fig. 8: Trajectory interpolation results on the overpass road.

Fig. 8 illustrates the trajectory interpolation results on the overpass road. The original trajectory is located along an intricate overpass that incorporates uphill and downhill segments and turns. As seen in Fig. 8, the results of PAR and GPR are not ideal; they produce only a rough shape with the true one. The two ensemble learning methods and RNN obtain similar trajectories, and the performance is good before the second turn. The RNN method shows up and down jitter as in other road sections. BPNN and OSVR produce the worst results. In contrast, our methods not only obtain clear trajectory but also exhibit a certain enhancement in the accuracy; however, after the second turn, the two trajectories predicted by linear and nonlinear R2C-TrA generate offset, which is triggered by longer outages and the inevitable accumulation of errors in this complex road section.

Table II presents the RMSE results, i.e., the average accumulated errors of the reconstructed trajectories. The nonlinear version of the proposed framework (nonlinear R2C-TrA) takes the polynomial kernel function as an example, and the experimental results are given accordingly. From the results in Table II, the proposed methods clearly outperform the existing regression methods such as GPR and BPNN; this superiority benefits from the effect of ensemble learning. Compared with the existing representative RNN of sequence prediction, the proposed method is more suitable for different application scenarios, thus it can obtain more stable experimental results. Moreover, the R2C-TrA framework outperforms the ensemble learning methods because the transfer learning process in our framework minimizes the sample weights of training samples that would be harmful to the current scenario. For the incremental learning methods, the effects on simple road sections such as fast and slow lanes are similar to the effect of the proposed framework, especially the linear framework. In complex road sections involving overpasses or turns, the incremental learning methods are inferior to our framework; the main reason is that incremental learning is good at processing varying and continuous data but is not sensitive to data fusion. In summary, our proposed methods employ the advantages of transfer learning in applications of vehicle trajectory interpolation tasks, and the experimental results are superior to those of the comparative methods.

In addition, we conduct the ablative study so as to to verify the proposed method more comprehensively. The results of the ablative study are shown in Table III. We select three road sections, the slow lane, the turning and the overpass in the experiments. The metric is RMSE. It can be seen from the experimental results that TrA-R2C (t=1) is the most unsatisfactory because the proposed model only works as a traditional SVR when t=1. Other cases, such as without incremental training (no-incremental), without data filtering (no-filtering) and lack of R2C process (no-R2C), have a quite similar RMSEs in each application scenario. The prediction accuracy of the proposed method in each application scenario are better than those cases. This is because the mutual assistance of each module is more helpful to solve the problem of trajectory interpolation when GNSS outages.

TABLE III: The performances of each component (RMSE)

Methods	**RMSE (meter)**			
	Slow-lane	Turning	Overpass	**Mean**
R2C-TrA(no-incremental)	6.99	40.10	43.36	30.15
R2C-TrA(no-filtering)	8.19	48.27	46.88	34.45
R2C-TrA(no-R2C)	9.08	45.20	54.67	36.32
R2C-TrA(t=1)	18.55	70.01	80.95	56.50
Linear R2C-TrA	4.01	33.90	28.19	22.03
Nonliner R2C-TrA	1.57	13.84	23.66	13.02

V. CONCLUSIONS

In this paper, we have developed the trajectory interpolation method with the aim at ensuring the availability of vehicle trajectory data when GNSS data become unavailable. A unified regression framework for integrated ensemble learning and transfer learning is proposed. First, the fusion of GNSS data and motion information is carried out. Then, the R2C framework is used to implement an incremental training process by building a more accurate classification-type loss

function for dataset learning. Furthermore, to account for missing trajectories, TrAdaboost is employed to construct a data filtering strategy that enhances the model accuracy. The proposed method is tested on real-world vehicle trajectory datasets, and the results show that the proposed framework is more accurate than traditional regression methods, including ensemble and incremental learning methods, on different road sections. However, we also find that on complex roads such as overpasses and turns, the proposed method cannot provide trajectories that fully overlap with the actual road because the long-term accumulation of deviations is inevitable. Overall, the proposed method is still an improvement because it incorporates transfer learning to correct errors as much as possible.

We have provide an open access for the collected vehicle trajectory data. Please see in https://github.com/HunanUniversityZhuXiao/PrivateCarTrajectoryData. In the future, we will combine the proposed framework with an online and multitask transfer learning approach to solve the trajectory interpolation problem of multiple data sources in more challenging scenarios, such as tunnels under rivers.

REFERENCES

[1] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, 2019.

[2] Z. Xiao, X. Dai, H. Jiang, D. Wang, H. Chen, L. Yang, and F. Zeng., " Vehicular Task Offloading via Heat-Aware MEC Cooperation Using Game-Theoretic Method ," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2038–2052, 2020.

[3] Y. Huang, Z. Xiao, D. Wang, H. Jiang, and D. Wu., "Exploring Individual Travel Patterns Across Private Car Trajectory Data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5036–5050, 2020.

[4] A. L. Alfeo, M. G. C. A. Cimino, S. Egidi, B. Lepri, and G. Vaglini, "A stigmergy-based analysis of city hotspots to discover trends and anomalies in urban transportation usage," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2258–2267, 2018.

[5] Z. Xiao, S. Xu, T. Li, R. Zhang, H. Jiang, A. C. Regan, and H. Chen., " On Extracting Regular Travel Behavior of Private Cars Based on Trajectory Data Analysis ," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 537–14 549, 2020.

[6] N. Alam, A. Kealy, and A. G. Dempster, "Cooperative inertial navigation for gnss-challenged vehicular environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1370–1379, 2013.

[7] V. Havyarimana, D. Hanyurwimfura, P. Nsengiyumva, and Z. Xiao, "A novel hybrid approach based-srg model for vehicle position prediction in multi-gps outage conditions," *Information Fusion*, vol. 41, pp. 1–8, 2018.

[8] Y. Huang, Z. Xiao, X. Yu, D. Wang, and J. Bai, "Road network construction with complex intersections based on sparsely sampled private car trajectory data," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 3, pp. 1–28, 2019.

[9] M. B. Younes and A. Boukerche, "A performance evaluation of a fault-tolerant path recommendation protocol for smart transportation system," *Wireless Networks*, vol. 24, no. 2, pp. 345–360, 2018.

[10] Z. Z. M. Kassas, M. Maaref, J. J. Morales, J. J. Khalife, and K. Shamei, "Robust vehicular localization and map matching in urban environments through imu, gnss, and cellular signals," *IEEE Intelligent Transportation Systems Magazine*, vol. 12, no. 3, pp. 36–52, 2020.

[11] M. Maaref and Z. M. Kassas, "Ground vehicle navigation in gnss-challenged environments using signals of opportunity and a closed-loop map-matching approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 7, pp. 2723–2738, 2020.

[12] X. Dong, C. Wang, Q. Yang, and W. Si, "System identification of distributed parameter system with recurrent trajectory via deterministic learning and interpolation," *Nonlinear Dynamics*, vol. 95, no. 1, pp. 73–86, 2019.

[13] V. Havyarimana, Z. Xiao, A. Sibomana, D. Wu, and J. Bai, "A Fusion Framework based on Sparse Gaussian-Wigner Prediction for Vehicle Localization using GDOP of GPS Satellites," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 680–689, 2020.

[14] V. Havyarimana, D. Hanyurwimfura, P. Nsengiyumva, and Z. Xiao, "A novel hybrid approach based-srg model for vehicle position prediction in multi-gps outage conditions," *Information Fusion*, vol. 41, pp. 1–8, 2018.

[15] Z. Xiao, F. Li, R. Wu, H. Jiang, Y. Hu, J. Ren, C. Cai, and A. Iyengar., "Trajdata: On vehicle trajectory collection with commodity plug-and-play obu devices," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 9066–9079, 2020.

[16] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, "Vehicle trajectory prediction by integrating physics- and maneuver-based approaches using interactive multiple models," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5999–6008, 2018.

[17] Y. Xing, C. Lv, and D. Cao, "Personalized vehicle trajectory prediction based on joint time-series modeling for connected vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1341–1352, 2020.

[18] H. Cheon and B. K. Kim, "Online bidirectional trajectory planning for mobile robots in state-time space," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4555–4565, 2019.

[19] A. Fachantidis, I. Partalas, M. E. Taylor, and I. Vlahavas, "Transfer learning with probabilistic mapping selection," *Adaptive Behavior*, vol. 23, no. 1, pp. 3–19, 2015.

[20] K. Duh and A. Fujino, "Flexible sample selection strategies for transfer learning in ranking," *Information Processing & Management*, vol. 48, no. 3, pp. 502–512, 2012.

[21] C. Ma, J. Xue, Y. Liu, J. Yang, Y. Li, and N. Zheng, "Data-driven state-increment statistical model and its application in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3872–3882, 2018.

[22] D. H. Jeong and J. M. Lee, "Ensemble learning based latent variable model predictive control for batch trajectory tracking under concept drift," *Computers & Chemical Engineering*, vol. 139, p. 106875, 2020.

[23] J. Xiao, Z. Xiao, D. Wang, J. Bai, V. Havyarimana, and F. Zeng, "Short-term traffic volume prediction by ensemble learning in concept drifting environments," *Knowledge-Based Systems*, vol. 164, pp. 213 – 225, 2019.

[24] S. Jiang, H. Mao, Z. Ding, and Y. Fu, "Deep decision tree transfer boosting," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 2, pp. 383–395, 2020.

[25] Z. Xiao, D. Xiao, V. Havyarimana, H. Jiang, D. Liu, D. Wang, and F. Zeng, "Towards accurate vehicle state estimation under non-gaussian noises," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 652–10 664, 2019.

[26] A. Noureldin, T. B. Karamat, M. D. Eberts, and A. El-Shafie, "Performance enhancement of mems-based ins/gps integration for low-cost navigation applications," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1077–1096, 2009.

[27] W. Dong, J. Liao, X. Zhu, X. Li, and V. Havyarimana, "Online-svr for vehicular position prediction during gps outages using low-cost ins," in *IEEE International Symposium on Personal*, 2015, pp. 1945–1950.

[28] Z. Xiao, P. Li, V. Havyarimana, H. M. Georges, D. Wang, and K. Li, "Goi: A novel design for vehicle positioning and trajectory prediction under urban environments," *IEEE Sensors Journal*, vol. 18, no. 13, pp. 5586–5594, 2018.

[29] S. Kumar, J. Paefgen, E. Wilhelm, and S. E. Sarma, "Integrating on-board diagnostics speed data with sparse gps measurements for vehicle trajectory estimation," in *Sice Conference*, 2013.

[30] S. H. Chen, J. S. Pan, and K. Lu, "Driving behavior analysis based on vehicle obd information and adaboost algorithms," *lecture notes in engineering & computer science*, 2015.

[31] N. El-Sheimy, K. W. Chiang, and A. Noureldin, "The utilization of artificial neural networks for multisensor system integration in navigation and positioning instruments," *IEEE Transactions on Instrumentation & Measurement*, vol. 55, no. 5, pp. 1606–1615, 2006.

[32] A. Sarkar, K. Czarnecki, M. Angus, C. Li, and S. Waslander, "Trajectory prediction of traffic agents at urban intersections through learned interactions," in *IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 1–8.

[33] X. Zhang, Z. Zhao, Y. Zheng, and J. Li, "Prediction of taxi destinations using a novel data embedding method and ensemble learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–11, 2019.

[34] J. Liang, L. Jiang, K. Murphy, T. Yu, and A. Hauptmann, "The garden of forking paths: Towards multi-future trajectory prediction," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 508–10 518.
[35] L. Wang, L. Zhang, and Z. Yi, "Trajectory predictor by using recurrent neural networks in visual tracking," *IEEE Transactions on Cybernetics*, vol. PP, no. 10, pp. 1–12, 2017.
[36] M. F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, and D. a. Ramanan, "Argoverse: 3d tracking and forecasting with rich maps," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8748–8757.
[37] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, and B. a. Caine, "Scalability in perception for autonomous driving: Waymo open dataset," pp. 2446–2454, 2020.
[38] J. Chen, Z. Xiao, D. Wang, W. Long, J. Bai, and V. Havyarimana, "Stay time prediction for individual stay behavior," *IEEE Access*, vol. 7, pp. 130 085–130 100, 2019.
[39] M. Muzammal, M. Gohar, A. U. Rahman, Q. Qu, A. Ahmad, and G. Jeon, "Trajectory mining using uncertain sensor data," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.
[40] J. Wang, C. Wang, X. Song, and V. Raghavan, "Automatic intersection and traffic rule detection by mining motor-vehicle gps trajectories," *Computers Environment & Urban Systems*, vol. 64, pp. 19–29, 2017.
[41] J. Xiao, Z. Xiang, D. Wang, and Z. Xiao, "Nonparametric kernel smoother on topology learning neural networks for incremental and ensemble regression," *Neural Computing & Applications*, no. 3, pp. 1–13, 2017.
[42] M. M. Adankon and M. Cheriet, "Support Vector Machine," *Computer Science*, vol. 1, no. 4, pp. 1–28, 2002.
[43] M. Muhlbaier, A. Topalis, and R. Polikar, "Learn++.MT: A New Approach to Incremental Learning," *Lecture Notes in Computer Science*, vol. 3077, no. 4, pp. 52–61, 2004.
[44] W. Dai, Q. Yang, G. R. Xue, and Y. Yu, "Boosting for transfer learning," in *International Conference on Machine Learning*, 2007.
[45] P. Geurts, D. Ernst, and L. Wehenkel, *Extremely randomized trees*. Kluwer Academic Publishers, 2006.
[46] A. Salas, S. J. Roberts, and M. A. Osborne, "A variational Bayesian State-Space Approach to Online Passive-Aggressive regression," in *Journal of Physics Conference Series*, vol. 590, no. 1, 2015.
[47] C. Chen, X. Liu, H. H. Chen, M. Li, and L. Zhao, "A rear-end collision risk evaluation and control scheme using a bayesian network model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 1–21, 2018.

Jianhua Xiao received a B.E. from Jiangxi Normal University, China and a Ph.D. degree from Hunan University, China, in 2015 and 2020, respectively. She is currently an instructor at School of Computer Science and Engineering, Huaihua University, Hunan. Her research interests include intelligent transportation systems and machine learning for traffic data analysis.

Zhu Xiao (M'15-SM'18) received M.S. and Ph.D. degrees in communication and information systems from Xidian University, China, in 2007 and 2010, respectively. From 2010 to 2012, he was a research fellow at the Department of Computer Science and Technology, University of Bedfordshire, U.K. He is currently an associate professor at the College of Computer Science and Electronic Engineering, Hunan University, China. His research interests include mobile communications, wireless localization, the Internet of Vehicles, next-generation communications and heterogeneous networks. He is a senior member of the IEEE. He is currently an Associate Editor for the *IEEE Transactions on Intelligent Transportation Systems.*

Dong Wang (M'14) received M.E. and Ph.D. degrees on computer science from Hunan University, China. He is a Ph.D. director and a director of overseas graduate students in the College of Computer Science and Electronics Engineering, Hunan University. His main research interests are computer networks and vehicular multimedia networks.

Vincent Havyarimana received his B.S. degree in mathematics from the University of Burundi, Bujumbura, in 2007, and his M.E. and Ph.D. degrees in computer science and electronic engineering from Hunan University, Changsha, China, in 2011 and 2016, respectively. He is currently an Associate Professor and Head of Section of Computer Engineering and Coordinator of Pedagogical Training Project with Ecole Normale Superieure, Burundi. His research interests include wireless positioning, vehicular ad hoc networks, and mobile computing.

Chenxi Liu received a B.S. degree from Southwest Minzu University, China, in 2018. She is currently pursuing a Ph.D. degree at Hunan University, China. Her main research interests include spatiotemporal data mining and graph data mining.

Chengming Zou received the Ph.D. degree from the Wuhan University of Technology, Wuhan, China, in 2003. He was a Visiting Research Scholar with the University of York, York, U.K., in 2008. He is currently a Professor with the School of Computer Science and Technology, Wuhan University of Technology. His research interests include the Internet of Things, artificial intelligence, and embedded systems.

Di Wu received a Ph.D. degree in computer science from the University of California, Irvine, USA, in 2013. He was a Researcher at the Intel Collaborative Research Institute for Sustainable Connected Cites, a Research Associate at Imperial College London, a Staff Research Associate at the University of California, Irvine, a Visiting Researcher at IBM Research, and a Student Research Associate at SRI International. He is currently a Professor at Hunan University, China, and an Adjunct Researcher at the University of California, Irvine, USA. His research interests include future networking, intelligent analytics, and smart architecture. He has actively served on many conference committees and is currently an Associate Editor for the *IEEE Transactions on Intelligent Transportation Systems*.