sensors

Article

Vehicle State Estimation Combining Physics-Informed Neural Network and Unscented Kalman Filtering on Manifolds

Chenkai Tan [1], Yingfeng Cai [1,*], Hai Wang [2], Xiaoqiang Sun [1] and Long Chen [1]

[1] Automotive Engineering Research Institute, Jiangsu University, Zhenjiang 212013, China; 2112104008@stmail.ujs.edu.cn (C.T.)
[2] School of Automotive and Traffic Engineering, Jiangsu University, Zhenjiang 212013, China
[*] Correspondence: caicaixiao0304@126.com

Abstract: This paper proposes a novel vehicle state estimation (VSE) method that combines a physics-informed neural network (PINN) and an unscented Kalman filter on manifolds (UKF-M). This VSE aimed to achieve inertial measurement unit (IMU) calibration and provide comprehensive information on the vehicle's dynamic state. The proposed method leverages a PINN to eliminate IMU drift by constraining the loss function with ordinary differential equations (ODEs). Then, the UKF-M is used to estimate the 3D attitude, velocity, and position of the vehicle more accurately using a six-degrees-of-freedom vehicle model. Experimental results demonstrate that the proposed PINN method can learn from multiple sensors and reduce the impact of sensor biases by constraining the ODEs without affecting the sensor characteristics. Compared to the UKF-M algorithm alone, our VSE can better estimate vehicle states. The proposed method has the potential to automatically reduce the impact of sensor drift during vehicle operation, making it more suitable for real-world applications.

Keywords: IMU calibration; unscented Kalman filtering on manifolds; physics-informed neural network; vehicle state estimation; multi-sensor fusion

check for updates

Citation: Tan, C.; Cai, Y.; Wang, H.; Sun, X.; Chen, L. Vehicle State Estimation Combining Physics-Informed Neural Network and Unscented Kalman Filtering on Manifolds. *Sensors* **2023**, *23*, 6665. https://doi.org/10.3390/s23156665

Academic Editors: Zahir M. Hussain and Hossam A. Gabbar

Received: 7 June 2023
Revised: 12 July 2023
Accepted: 21 July 2023
Published: 25 July 2023

1. Introduction

The rapid development of sensor technology and intelligent transportation systems [1] in recent years has led to the introduction of new vehicle chassis subsystems by original equipment manufacturers [2]. These subsystems improve specific vehicle performance, making driving more efficient and effective. Vehicle chassis subsystems not only make driving more convenient [3], but they also make the vehicle a complex autonomous system. The optimal coordination of chassis system (OCCS) [4] is applied to coordinate different complementary chassis subsystems. However, accurate vehicle state information is required for chassis coordinated control in order to correctly coordinate subsystems and identify driving conditions [5]. Furthermore, sensor noise can affect the accuracy, reliability, and continuity of vehicle state information. Extensive research [6,7] has been conducted to investigate various estimation algorithms based on cost-effective sensors and available measurements.

With the development of sensor technology, the use of vehicle control sensors such as the global navigation satellite system (GNSS) and the inertial measurement unit (IMU) has increased. In the OCCS, the IMU plays a crucial role in measuring the angular rate and acceleration of the vehicle body, enabling accurate calculations of the vehicle's attitude, velocity, and position [8]. However, due to installation errors [9] or coupling effects between the IMU and vehicle motion [10], IMU misalignment is inevitable. Indirect state estimation methods [9,10] are proposed to mitigate the drift of IMUs. To achieve advanced control, such as the OCCS, the coordinator requires multiple advanced sensor inputs and more estimation outputs, primarily to avoid conflicts downstream in subsystems [2]. This has motivated us to develop a method that leverages the relationships between advanced sensors to eliminate IMU drift.

The extended Kalman filter (EKF [11]) and unscented Kalman filter (UKF [12]) are commonly used for optimal integration between the GNSS and inertial navigation system (INS). The unscented Kalman filter on manifolds (UKF-M) [13] is a novel filtering algorithm that provides more accurate and robust navigation estimates. Compared with the existing state-of-the-art integrated navigation algorithms, the UKF-M-based integrated navigation estimation algorithm [14] has higher accuracy and faster convergence speed. These methods incorporate various vehicle/tire models and real-time states [15] to handle disturbances and noise. However, the local linearization operation of EKF/UKF can introduce significant estimation errors [7]. Additionally, sensor offset can cause integration errors in the INS/GNSS model [16].

A virtual sensor (VS) [17] can be used to replace a redundant sensor, which can mitigate sensor noise. A VS is a type of software sensor that can integrate multiple data sources to improve system reliability. Kim et al. [18] combined the adaptive Kalman filter with a deep neural network (DNN) to estimate the sideslip angle. The proposed model utilized the DNN output as a VS. Combining the EKF/UKF model with the DNN-based VS, this model not only provided accurate estimates of the sideslip angle but also quantified the uncertainty associated with the estimation. In another study, Kim et al. [19] used a long short-term memory (LSTM) network to filter the noise and bias of the original sensor data. Leandro et al. [20] combined a neural network and a Kalman filter to estimate the vehicle's roll angle. The neural network output was used as the pseudo-roll angle to build the Kalman module. Soriano et al. [21] proposed a neural network-based calibration method for a two-axis accelerometer. Their experimental results demonstrated that the accelerometer error model based on the neural network had better accuracy and robustness than the explicit accelerometer error model method.

The data-driven vehicle model [18–20] serves as an approach for establishing VS. These models utilize a data-driven approach to estimate vehicle states by leveraging the hidden relationships between them. In the development of data-driven vehicle dynamics models, the linear time-invariant (LTI) state-space model [22–25] is commonly employed. Experimental results [23–25] demonstrated that the LTI-based data-driven vehicle model outperformed comparative vehicle dynamics models. However, these data-driven vehicle models [18–20,23–25] rely on the assumption that the selected supervised learning vehicle states accurately represent the true vehicle states. This assumption can impact the effectiveness of online learning in these models. Additionally, due to the influence of sensor noise [26], neural network-based models may introduce errors without physics-based models.

The physics-informed neural network (PINN) [27] is a novel deep learning method that integrates domain-specific knowledge into a neural network architecture. Xu et al. [28] proposed a PINN-based model for unmanned surface vehicle dynamics. Compared with traditional neural networks, their PINN-based unmanned surface vehicle dynamics model had better prediction accuracy for the sway and surge velocity and rotation speed. Franklin et al. [29] used PINN as a hybrid virtual sensor to estimate the flow metering in oil wells. Wong et al. [30] demonstrated that PINN can effectively mitigate the impact of noise in data originating from low-quality sensors. In state estimation, combining PINN with a state-space model formulation can avoid computationally costly time integration [31].

In this study, we propose a vehicle state estimation (VSE) method that combines PINN and UKF-M (PINN UKFM). The contributions of the paper can be divided into two main parts.

- We use PINN as a data-driven vehicle dynamics model to establish a VS, using the IMU calibration values as the output. By leveraging the loss calculation method of PINN, the vehicle dynamics can be integrated with the data-driven model, enabling the incorporation of information from multiple sensor sources during vehicle operation. The experimental results in a real vehicle platform indicated that the PINN-based model effectively integrates multiple sensor inputs to achieve improved estimation of the vehicle’s state, surpassing both the physical and neural network-based models.

- Based on the IMU calibration values, we utilize the UKF-M algorithm to estimate the altitude, velocity, and position of the vehicle. By fusing data from multiple sensors, PINN UKFM provided accurate and comprehensive vehicle states that included six-dimensional vehicle dynamics, 3D attitude, speed, and position, which can be used in various vehicle dynamic control systems. For example, the six-dimensional vehicle dynamics can define the chassis motion, which can be used in OCCS. The 3D position can be applied to vehicle navigation in a GNSS-denied environment. The experimental results indicated that the PINN-based model can effectively incorporate multiple sensor inputs to mitigate IMU biases and enhance the accuracy of the existing state-of-the-art integrated navigation algorithm, UKF-M.

The rest of work is organized as follows: Section 2 introduces the vehicle model, the PINN UKFM sensor states, and defines the PINN UKFM problem. Next, Section 3 introduces the proposed PINN UKFM algorithm. In Section 4, the proposed method is tested using realistic vehicle data. Rainy weather is used as an experimental condition as it reduces the road adhesion coefficient and increases the nonlinearity of vehicle dynamics. Finally, Section 5 presents the conclusions of this paper.

2. Estimation Problem

2.1. The Vehicle Model

The PINN UKFM algorithm used a six-degrees-of-freedom (6-DoF) kinematic vehicle model [32] to obtain accurate state estimates. As shown in Figure 1, the model could define the chassis movement [33], which included the navigation and vehicle body coordinates [5]. The starting point of the navigation coordinates was defined as the track start. The navigation coordinates consisted of three variables: E (east), N (north), and U (upward).

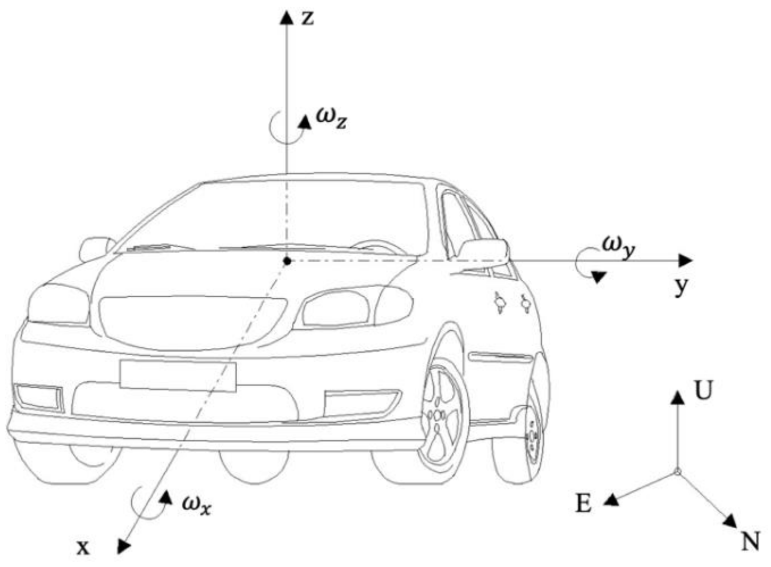

Figure 1. The 6-DoF vehicle model coordinates.

The vehicle body's coordinate origin point was located at its center of mass and the right-hand rule was used. The x-, y-, and z-directions pointed forward, left, and upward, respectively. Acceleration and velocity could be broken down into longitudinal acceleration a_x/velocity v_x, lateral acceleration a_y/velocity v_y, and vertical acceleration a_z/velocity v_z. The vehicle's direction angle was defined as the rolling angle (around x, roll rate ω_x), pitching angle (around y, pitch rate ω_y), and heading angle (around z, yaw rate ω_z). Finally, other vehicle states could be extrapolated from these. For example, the sideslip angle β could be calculated through the following trigonometric operation:

$$\beta = \tan^{-1}\left(\frac{v_y}{v_x}\right) \tag{1}$$

2.2. The Sensor States

In the experiment, the vehicle had multiple sensors, including the GNSS, IMU, measurement steering wheels (MSW), wheel force transducers (WFT), 2-axis optical sensors (S-Motion), and a monocular camera. The experimental platform structure is presented in Section 4.1. Due to the real-time computation requirement, the signals from the monocular camera were not used in the PINN UKFM algorithm. The sensor inputs of PINN UKFM are introduced in Table 1.

Table 1. The sensor inputs of PINN UKFM.

Sensor Types	Signal Name	Symbol	Units
GNSS	Easting	E	m
GNSS	Northing	N	m
GNSS	Altitude	U	m
GNSS	UTM velocity	V_{gps}	m/s
IMU	Roll angle	φ	rad
IMU	Pitch angle	Θ	rad
IMU	Yaw angle	ψ	rad
IMU	Vertical velocity	v_z	m/s
S-Motion	Longitudinal velocity	v_x	m/s
S-Motion	Longitudinal acceleration	a_x	m/s^2
S-Motion	Lateral velocity	v_y	m/s
S-Motion	Longitudinal acceleration	a_y	m/s^2
S-Motion	Vertical acceleration	a_z	m/s^2
S-Motion	Roll rate	ω_x	rad/s
S-Motion	Pitch rate	ω_y	rad/s
S-Motion	Yaw rate	ω_z	rad/s
WFT	Wheel force	F_x, F_y, F_z	kN
WFT	Wheel torque	M_x, M_y, M_z	kN·m
MSW	Steering wheel angle	δ_s	rad

The symbols E and N represented easting and northing in the navigation coordinates. Using the universal transverse mercator (UTM) and the navigation starting point, easting and northing were transformed from the GNSS coordinates into the navigation coordinates. The UTM velocity was calculated from navigation coordinates as:

$$v_{gps} = \frac{\sqrt{\left(E^t - E^{t-1}\right)^2 + \left(N^t - N^{t-1}\right)^2 + \left(U^t - U^{t-1}\right)^2}}{dt} \tag{2}$$

where dt is a single GNSS interval, and t and $t-1$ are GNSS coordinate timestamps.

2.3. Problem Definition

The PINN-based VS was defined as a time-series forecasting model in which sensor signals were taken as discrete variables. Assuming that "n" represents the current timestamp, the PINN module input states were defined as:

$$\begin{aligned} X &= [X_{n-L}, X_{n-L+1}, \ldots, X_n] \\ X_n &= \left[X_n^{(1)}, X_n^{(2)}, \ldots, X_n^{(\vartheta)}\right] \end{aligned} \tag{3}$$

where $n - L$ is the starting time step, X represents the sensor signals shown in Table 1, and ϑ is the number of sensor signals.

PINN was used as a universal function approximator to achieve IMU calibration. Building upon previous artificial intelligence-based techniques and the integration of the Kalman filter for estimation [23], the calibrated values were called “pseudo-states.” By applying the conservation principles derived from the vehicle dynamics, the pseudo-states satisfied the conservation principles originating from the vehicle dynamics. Therefore, the PINN module output states were defined as:

$$\hat{u}_\theta = \left[\hat{u}_\theta^{(1)}, \hat{u}_\theta^{(2)}, \dots, \hat{u}_\theta^{(\iota)}\right] \tag{4}$$

where $\hat{u}_\theta$ represents the pseudo-states, and ι is the number of pseudo-states. Based on the LTI state space assumption, the pseudo-states were inputted to ODEs to compute the integrated states. The sensor measurements of these integrated states z were represented as:

$$\begin{aligned} z &= [z_{n+1}, z_{n+2}, \dots, z_{n+F}] \\ z_{n+1} &= \left[z_{n+1}^{(1)}, z_{n+1}^{(2)}, \dots, z_{n+1}^{(\kappa)}\right] \end{aligned} \tag{5}$$

where $[n+1, n+2, \dots, n+F]$ represent the output timestamps, κ is the number of the integrated states, and $n + F$ is the ending time step.

For better integration with the vehicle control, the authors hypothesized that the vehicle physical model satisfied an LTI state-space model [23,24], which could be defined as:

$$\begin{cases} \dot{x} = Ax_t + \mathscr{B}_t \\ y_t = Cx_t + \mathscr{D}_t \end{cases} \tag{6}$$

where x_{t+1} is the state variable at next time step, x_t is the state variable at current time step, y_t represents the output variable at current time step, $\mathscr{B}_t$and$\mathscr{D}_t$ are disturbances or noise, and $\dot{x}$ denotes the rate of change of the state variable. It should be noted that the system dynamics and output equations did not change over $[n+1, n+2, \dots, n+F]$.

By utilizing the LTI state-space model, the output of the PINN module could be used to compute the other vehicle states. Based on the ODE constraints, PINN ensured the estimated IMU states satisfied the physical relationships among sensor measurements.

Next, the pseudo-states were inputted into the UKF-M-based IMU-GNSS sensor-fusion model [34]. Using these pseudo-states, the UKF-M module could estimate both the velocity and position of the vehicle.

3. Methodology

3.1. Structure of the Proposed VSE

In recent years, there has been growing interest in the development of multi-sensor systems for vehicle state estimation due to their potential to improve accuracy and robustness in complex environments. The proposed PINN UKFM is one such system, integrating multiple sensors to estimate the vehicle dynamics in real time. The architecture of our proposed PINN UKFM is illustrated in Figure 2 and consisted of three modules: the sensors module, the PINN module, and the UKF-M module.

The sensors module in PINN UKFM consisted of various sensors, such as GNSS, IMU, MSW, WFT, and S-Motion. These sensors provide rich information about the vehicle's motion, including its position, velocity, acceleration, and orientation. For adaptive reduction of data noise, the PINN module was employed to learn the complex, nonlinear relationship between the sensor signals and the filtered vehicle states. Specifically, the PINN module used the time-series sensor signals as input and outputted the corresponding pseudo-states. These pseudo-states were then fed into the UKF-M module, which used a state-space model to estimate the vehicle's position, velocity, and other parameters. By combining the strengths of both PINN and UKF-M, the proposed PINN UKFM achieved accurate and robust vehicle state estimation.

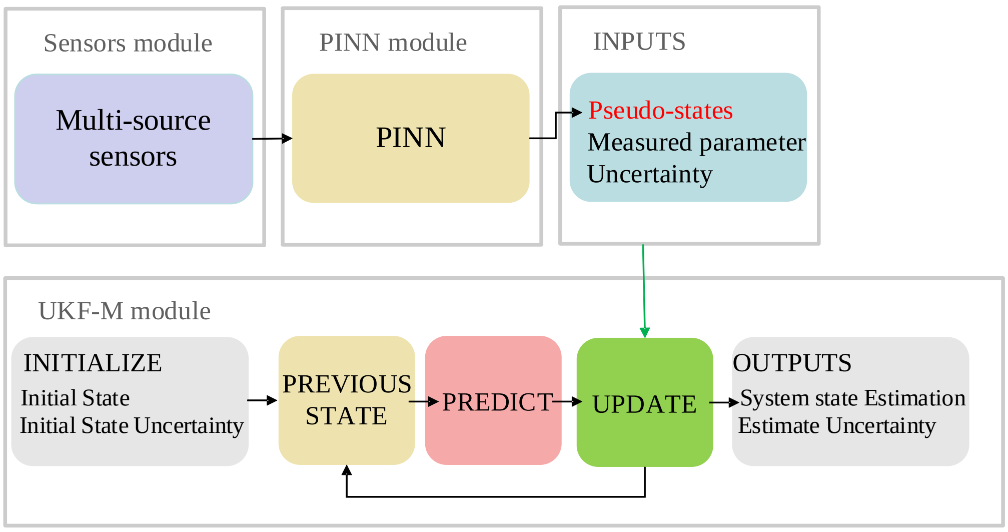

Figure 2. The structure of the proposed VSE.

3.2. The PINN Module

In practical applications, different sensors may measure data with noise and drift due to their different characteristics and working environments. To obtain accurate vehicle state estimation, we used PINN to integrate the vehicle dynamics into a neural network architecture. The PINN module punished the loss function with ODEs and algebraic equations to make the sensor data consistent with the vehicle dynamics. The PINN module fused the data from multiple sensors to reduce measurement errors and make the data more consistent with the actual vehicle dynamics.

Therefore, the PINN module found the angle and velocity by integrating the acceleration and angular velocity. Temporal interaction is widely used in establishing data-driven vehicle models [19,25], as it can capture the complex temporal and hidden correlations for better state prediction. Therefore, a temporal model consisting of an encoder layer, a temporal interaction layer, and a decoder layer was proposed, as shown in Figure 3.

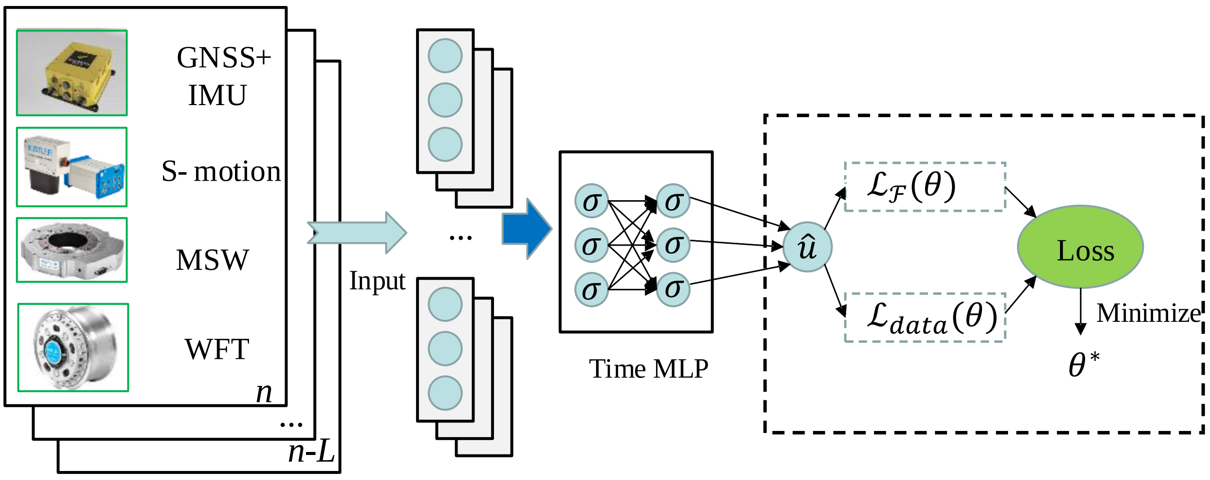

Figure 3. The proposed PINN module.

The encoder layer was used to embed and encode sensor signal X. Data were mapped into the high dimension through the multilayer perceptron (MLP). The encoder layer was defined as:

$$e_t = \sigma(W_{emb} X_t + b_{emb}) \tag{7}$$

where e_t denotes the embedded feature vector. Additionally, the MLP contained the weighting matrix W_{emb}, bias term b_{emb}, and the rectified linear unit function (ReLU) activation function σ [35].

Since vehicle states have temporal interactions, the temporal interaction layer supposed the temporal interactions of different hidden states. The time dimension of e_t was connected:

$$e^0 = \text{Concat}(e_t), t \in [n - L, n] \tag{8}$$

where the embedded feature vectors were concatenated to form e^0. Then, the PINN module learned the temporal interaction:

$$e^l = \sigma\left(W^l_{time} e^{l-1} + b^l_{time}\right),\ l = 1,\ 2,\ \ldots, s \tag{9}$$

where e^{l-1} and e^l are the input and output of layer l, respectively.

The decoder layer predicted the pseudo-states as:

$$\hat{u}_\theta = (W_{dec} e^s + b_{dec}) + u_{past} \tag{10}$$

where u_{past} represents the past measurement of the IMU, and $\hat{u}_\theta$ refers to the pseudo-states that represent the IMU calibration values. We defined the residual between the pseudo-states and past measurement of the IMU as the drift of the IMU. This structure was similar to the residual block in a residual network [36]. $(W_{dec} e^s + b_{dec}) = \hat{u}_\theta - u_{past}$ represented the latent (hidden) solution of the drift of IMU. The PINN determined the parameter θ of the NN [27] by minimizing the loss function:

$$\begin{gathered}
\theta = \text{argmin}\ \mathscr{L}(\theta) \\
\mathscr{L}(\theta) = \mathscr{L}_{\mathcal{F}}(\theta) + \mathscr{L}_{data}(\theta) \\
\mathscr{L}_{data}(\theta) = \frac{1}{N_d} \sum_{i=1}^{N_d} |\hat{u}_\theta(X;\theta) - u(X)|^2 \\
\mathscr{L}_{\mathcal{F}}(\theta) = \sum_{k=1}^{7} \omega_k \cdot \left[\frac{1}{F}\frac{1}{N_{\mathcal{F}}} \sum_{t=n+1}^{n+F} \sum_{i=1}^{N_{\mathcal{F}}} |g_k(\hat{u}_\theta(X;\theta), X_n, z_t)|^2\right] + \sum_{k=8}^{12} \omega_k \cdot \left[\frac{1}{N_{\mathcal{F}}} \sum_{i=1}^{N_{\mathcal{F}}} |f_k(\hat{u}_\theta(X;\theta), X_n)|^2\right]
\end{gathered} \tag{11}$$

where $\mathscr{L}_{\mathcal{F}}(\theta)$ represents the mean square error of residuals of the physics-based equations; $\mathscr{L}_{data}(\theta)$ represents the mean square error of residuals of the measurement data; $\omega_1, \omega_2, \ldots, \omega_{12}$ denote the weights associated with physical constraints; N_d and $N_{\mathcal{F}}$ represent the batch size; $u(X)$ represents the future measurement of the IMU; t represents the output timestamps; F represents the forecast horizon; $g(\hat{u}_\theta(X;\theta), X_n, t)$ represents the ordinary differential equations [37]; and $f(\hat{u}_\theta(X;\theta), X_n, t)$ represents the algebraic equations [37]. The ODEs and algebraic equations were utilized as an additional regularization term [38]. By minimizing the loss function of the physics-based equations, we could incorporate the laws of physics into the NN [38–40].

When used for computing partial differential equations (PDE), the $\mathscr{L}_{\mathcal{F}}(\theta)$ in PINN is typically used to penalize the degree to which the model violates physical laws. However, the PDE-based $\mathscr{L}_{\mathcal{F}}(\theta)$ was not applicable in the continuous time modeling and prediction problems addressed in this paper. The physical laws of vehicle dynamics models are often subject to ODEs and algebraic equations. To address this, we drew inspiration from the approach of a physics-constrained neural network (PCNN) [39,40] and neural ordinary differential equations (NODEs) [41,42]. PCNN is a specific variant of PINN that introduces a regularization parameter to control the trade-off between data and knowledge-based regularization [43]. In NODEs, the hidden layers of a neural network are treated as the states of an ODE, and an ODE solver is used to compute the evolution of these states.

Specifically, we incorporated ODEs to represent the dynamic evolution of the system and used algebraic equations to represent the vehicle dynamics model.

The PINN module provided the signals of the pseudo-states, which represented the rate of change of the state variable $\dot{x}$ in the LTI state space. The inputs and outputs of the PINN module were denoted as:

$$\begin{gathered} X^{(1-22)} = \left[a_x, a_y, a_z, \omega_x, \omega_y, \omega_z, v_x, v_y, v_x, \varphi, \Theta, \psi, M_{y_l}, M_{y_r}, F_{x_{fl}}, F_{x_{rl}}, F_{y_{fl}}, F_{y_{rl}}, d_E, d_N, d_U, \delta_s\right] \\ \hat{u}_\theta(X;\theta)^{(1-6)} = \left[a_x, a_y, a_z, \omega_x, \omega_y, \omega_z\right] \end{gathered} \tag{12}$$

where d_E, d_N, and d_U are the navigation coordinates minus the current vehicle coordinates [44].

The ordinary differential equations were represented as:

$$\begin{gathered} g_q(\hat{u}_\theta(X;\theta), X_n, z_t) = X_n^{(q+6)} + \hat{u}_\theta(X;\theta)^{(q)} \times d_t - z_t^{(q)}, q = 1,\ 2,\ \ldots,\ 6 \\ g_7(\hat{u}_\theta(X;\theta), X_n, z_t) = \sum_{i=1}^{3} \left[X_n^{(i+6)} \times d_t + \frac{\hat{u}_\theta(X;\theta)^{(i)} \times {d_t}^2}{2}\right] - z_t^{(7)} \end{gathered} \tag{13}$$

where $X_n^{(7-12)} = \left[v_{x_n}, v_{y_n}, v_{z_n}, \varphi_n, \Theta_n, \psi_n\right]$ are the initial states of ODE outputs; $z_t^{(1-6)} = \left[v_{x_t}, v_{y_t}, v_{z_t}, \varphi_t, \Theta_t, \psi_t\right]$ are the measurements of ODE outputs at timestamp t; and d_t is the time interval between n and t. By minimizing $g_1 \sim g_6$, the pseudo-states could incorporate information from the related vehicle states. The state $z_t^{(7)}$ represents the position change of the vehicle, which was represented as:

$$\begin{gathered} z_t^{(8,9,10)} = [d_{E_t}, d_{N_t}, d_{U_t}] \\ z_t^{(7)} = \sqrt{(E^t - E^n)^2 + (N^t - N^n)^2 + (U^t - U^n)^2} = \sqrt{(d_{E_t})^2 + (d_{N_t})^2 + (d_{U_t})^2} \end{gathered} \tag{14}$$

where $[E, N, U]$ are the outputs of variable y_t in the LTI state-space vehicle model. By minimizing g_7, the physical knowledge of y_t was incorporated into the physics-informed loss function. The position-updated formula of the vehicle dynamics [45] may affect the learning effectiveness of the neural network. Therefore, we used the Euler integral to calculate the displacement of the vehicle. By utilizing the loss calculation method of PINN, the ODE vehicle dynamics could be combined with a data-driven model to consider multiple sensor sources.

Algebraic equations capture simple dependence relationships among vehicle states [37]. The algebraic equations representing the linear dynamics in the LTI state space could be written as $\dot{x} = AX_n + \mathscr{B}_n$. By minimizing the loss of $\hat{u}_\theta(X;\theta) - \dot{x}$, we could incorporate the physical dynamics model into the physics-informed loss function. The referenced vehicle dynamics model was based on the two-degree-of-freedom (2-DOF) vehicle dynamics model [23,45]. To simplify the vehicle dynamics [23], we decoupled the longitudinal and latitudinal dynamics by neglecting the influence of the latitudinal and longitudinal forces.

The algebraic equation of $f_8(\hat{u}_\theta(X;\theta), X_n)$ was represented as:

$$\begin{gathered} f_8(\hat{u}_\theta(X;\theta), X_n) = a_{x_n}^{(1)} - \hat{u}_\theta(X;\theta)^{(1)} \\ X_n^{(7,13,14)} = \left[v_{x_n}, M_{y_{l,n}}, M_{y_{r,n}}\right] \\ m a_{x_n}^{(1)} = F_{e_n} + F_{b_n} + F_{s_n} + F_{f_n} - F_{D_n} \\ a_{x_n}^{(1)} = \frac{1}{m}\left(k_1 \times M_{y_{l,n}} + k_2 \times M_{y_{r,n}} + mg\gamma + mg\mu - k_D v_{x_n}^2\right. \end{gathered} \tag{15}$$

where $a_{x_n}^{(1)}$ is calculated based on the vehicle longitudinal dynamics; F_{e_n} represents the transmitted force of the vehicle; F_{b_n} represents the brake force of the vehicle; m is the gross vehicle mass; and $M_{y_{l,n}}$ and $M_{y_{r,n}}$ represent the y-axis torque of the two front wheels. We assumed F_{e_n} and F_{b_n} were transmitted to the front wheels and converted into $M_{y_{l,n}}$ and $M_{y_{r,n}}$, respectively (the experimental car was a front-wheel drive vehicle). Therefore, the transmitted and brake forces were calculated using the least squares method. Additionally, a small slope angle was assumed. Therefore, the dissipative force was $F_{s_n} = 0$ and the frictional force was $F_{f_n} = mg\mu$. Here, g represented the gravitational constant and μ was the road–wheel static friction coefficient. The air drag force F_{D_n} was calculated based on the longitudinal velocity and the drag coefficient k_D. The parameters k_1, k_2, μ, and k_D were calculated using the least squares method to realize online learning of the parameters.

In this paper, the tire force was directly measured. This allowed for the direct calculation of the longitudinal acceleration $a_{x_n}^{(2)}$ from the measured tire force. The algebraic equation of $a_{x_n}^{(2)}$ was represented as:

$$\begin{aligned} f_9(\hat{u}_\theta(X;\theta), X_n) &= a_{x_n}^{(2)} - \hat{u}_\theta(X;\theta)^{(1)} \\ X_n^{(15,16)} &= \left[F_{x_{fl,n}}, F_{x_{fr,n}}\right] \\ a_{x_n}^{(2)} &= \frac{\beth_1}{m}\left(F_{x_{fl,n}} + F_{x_{fr,n}}\right) \end{aligned} \tag{16}$$

where $F_{x_{fl,n}}$ and $F_{x_{fr,n}}$ are the front left and right wheel longitudinal forces, respectively; $a_{x_n}^{(2)}$ represents the longitudinal acceleration, $\beth_1$ is the parameter ($\beth_1 = \frac{F_{x_{fl,1}} + F_{x_{fr,1}}}{ma_{x_1}}$), and $[F_{x_{fl,1}}, F_{x_{fr,1}}, a_{x_1}]$ represents the known measurement data.

The algebraic equation of $f_{10}(\hat{u}_\theta(X;\theta), X_n)$ was calculated from the vehicle latitudinal dynamics, which was represented as:

$$\begin{aligned} f_{10}(\hat{u}_\theta(X;\theta), X_n) &= a_{y_n}^{(1)} - \hat{u}_\theta(X;\theta)^{(2)} \\ X_n^{(7,8,12,22)} &= \left[v_{x_n}, v_{y_n}, \omega_{z_n}, \delta_{s_n}\right] \\ \delta_{f_n} &= \frac{1}{\mathrm{i}}\delta_{s_n} \\ ma_{y_n}^{(1)} &= F_{y_{fl,n}} + F_{y_{fr,n}} + F_{y_{rl,n}} + F_{y_{rr,n}} \\ a_{y_n}^{(1)} &= \frac{1}{m}\left[\frac{-2\mathrm{C}v_{y_n} - 2\mathrm{C}\left(l_f - l_r\right)\omega_{z_n}}{v_{x_n}} - mv_{x_n}\omega_{z_n} + 2\mathrm{C}\delta_{f_n}\right] \end{aligned} \tag{17}$$

where C represents the tire cornering stiffness; l_f and l_r represent the distances from the center of the vehicle's mass to the front and rear axles, respectively; ω_z denotes the yaw rate; $F_{y_{fl,n}}, F_{y_{fr,n}}, F_{y_{rl,n}}$, and $F_{y_{rr,n}}$ denote the lateral tire forces at the front left/right and rear left/right wheels, respectively; δ_{s_n} represents the steering wheel angle; δ_{f_n} represents the front wheel steering angle; and i denotes the function of the variable steering gear ratio [46]. This was a linear tire model and we assumed the vehicle had equal cornering stiffness for all four wheels.

Similar to $a_{x_n}^{(2)}$, we directly calculated the latitudinal acceleration $a_{y_n}^{(2)}$ from the measured tire force. The algebraic equation of $a_{y_n}^{(2)}$ was represented as:

$$\begin{aligned} f_{11}(\hat{u}_\theta(X;\theta), X_n) &= a_{y_n}^{(2)} - \hat{u}_\theta(X;\theta)^{(2)} \\ X_n^{(17,18)} &= \left[F_{y_{fl,n}}, F_{y_{rl,n}}\right] \\ a_{y_n}^{(2)} &= \frac{\beth_2}{m}\left(F_{y_{fl,n}} + F_{y_{fr,n}}\right) \end{aligned} \tag{18}$$

where $F_{y_{fl,n}}$ and $F_{y_{fr,n}}$ represent the front left and right wheel latitudinal forces, respectively; $a_{y_n}^{(2)}$ represents the latitudinal acceleration; and $\beth_2$ is the parameter ($\beth_2 = \frac{F_{y_{fl,1}} + F_{y_{fr,1}}}{ma_{y_1}}$).

The algebraic equation of $f_{12}(\hat{u}_\theta(X;\theta), X_n)$ was calculated from the kinematic vehicle model, which was represented as:

$$\begin{aligned} f_{12}(\hat{u}_\theta(X;\theta), X_n) &= \omega_{z_n}^{(1)} - \hat{u}_\theta(X;\theta)^{(6)} \\ \beta_n &= \tan^{-1}\left[\frac{l_r}{l_f + l_r}\tan\left(\delta_{f_n}\right)\right] \\ \omega_{z_n}^{(1)} &= \frac{v_{x_n}}{l_f} \times \sin(\beta_n) \end{aligned} \tag{19}$$

where β_n represents the vehicle slip angle.

The physics-based equations presented in this paper, ($g_{1-7}(\hat{u}_\theta(X;\theta), X_n, z_t)$ and $f_{8-12}(\hat{u}_\theta(X;\theta), X_n)$), were applicable within the LTI state space. This implied that the pseudo-states were assumed to remain constant over the time interval $[n+1, n+2, \ldots, n+\mathrm{F}]$. This assumption ensured the validity and applicability of the equations mentioned earlier, allowing for the incorporation of physics-based constraints into the neural network model.

The PINN module utilized the PyTorch deep learning framework. To consider the dynamics of different states, the Adam optimizer with a 0.001 learning rate was used to train the network. The model was trained using a Nvidia GTX 3080Ti GPU.

3.3. The UKF-M Module

The UKF-M is a novel UKF on manifolds, with versatility that allows direct application to numerous practical manifolds. For stochastic processes on Riemannian manifolds, the theory of Lie groups [47] is used to define the vehicle's attitude estimation. The IMU-GNSS sensor-fusion model [34] was a UKF-M-based filter, which is a standard 3D kinematic model based on inertial inputs. The UKF-M algorithm utilized in PINN UKFM was based on the methodology proposed in [34]. The modification of PINN UKFM involved using the outputs of the PINN module to replace the original IMU measurements.

The states of a moving vehicle in a discrete dynamic system are represented as:

$$\chi_n \in \mathscr{M} = \left\{ C_n \in \mathbb{R}^{3\times 3}, v_n \in \mathbb{R}^3, \mathscr{P}_n \in \mathbb{R}^3, b_{g_n} \in \mathbb{R}^3, b_{a_n} \in \mathbb{R}^3 \right\} \tag{20}$$

where χ_n denotes the state of a vehicle belonging to a parallelizable manifold $\mathscr{M}$; n is the current timestamp; $v_n = (v_{E_n}, v_{N_n}, v_{U_n})$ is the velocity vector (v_{E_n}–velocity east and v_{N_n}-velocity north); $\mathscr{P}_n = (E_n,\ N_n, H_n)$ is vehicle coordinate in the navigation coordinates; $b_{g_n} = \left(b_{\omega_{x,\ n}}, b_{\omega_{y,\ n}}, b_{\omega_{z,\ n}}\right)$ is the gyro bias; $b_{a_n} = \left(b_{a_{x,\ n}}, b_{a_{y,\ n}}, b_{a_{z,\ n}}\right)$ is the accelerometer bias; and C_n is a special orthogonal group that represents 3D rotation [47].

$$SO(3) := \left\{ C_n \in \mathbb{R}^{3\times 3} \middle| C_n {C_n}^{\mathrm{T}} = 1, \det C_n = 1 \right\} \tag{21}$$

where 1 is the identity matrix. Based on the time derivative of $C_n {C_n}^{\mathrm{T}} = 1$, a skew-symmetric matrix $C_n^T \dot{C}_n$ was obtained:

$${C_n}^{\mathrm{T}} \dot{C}_n + {\dot{C}_n}^{\mathrm{T}} C_n = 0 \tag{22}$$

The $C_n^T \dot{C}_n$ as a skew-symmetric matrix is often noted as $[\omega]_\times$:

$${C_n}^{\mathrm{T}} \dot{C}_n = [\omega]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{23}$$

where $[\omega]_\times$ is in the Lie algebra of $SO(3)$. The Lie algebra is a vector space and can be decomposed into:

$$[\omega]_\times = \omega_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} + \omega_y \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} + \omega_z \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{24}$$

where $\omega = \left[\omega_x, \omega_y, \omega_z\right]$ is in the vector of angular velocities. For the ω constant, we obtained the ODE solution:

$$C_n = \exp\left([\omega]_\times n\right) \tag{25}$$

where exp() is the exponential map on the SO(3) [42] and $C_0 = I$. The exp() map was derived from the time derivatives of $\chi_n \in \mathscr{M}$. The vector fields were defined as:

$$V_1(C_n) = C_n \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^{\wedge}, V_2(C_n) = C_n \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}^{\wedge}, V_3(C_n) = C_n \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^{\wedge} \tag{26}$$

where $\hat{}$ is the hat map [47], and V_1, V_2, and V_3 are the vector fields.

Similar to the Gaussian belief of the Kalman filter, the UKF-M algorithm builds a probability distribution as $\chi_n \sim \mathcal{N}(\hat{\chi}_n, P_n)$ for the random variable $\chi_n \in \mathscr{M}$ as:

$$\chi_n = \boldsymbol{\varphi}(\hat{\chi}_n, \xi_n), \xi_n \sim \mathcal{N}(0, \mathrm{P}_n) \tag{27}$$

where $\hat{\chi}_n$ is viewed as the mean estimate at timestep n; $\boldsymbol{\varphi}$ is the propagation function; $\xi_n \in \mathbb{R}^d$ is a random Gaussian vector; $\mathcal{N}$ is the Gaussian distribution; and $\mathrm{P}_n \in \mathbb{R}^{d\times d}$ is the covariance matrix. $\varphi(\hat{\chi}_n, \xi_n) \in \mathscr{M}$ is obtained by starting from $\hat{\chi}_n$ and integrating the vector field $\sum_{i=1}^{d} \xi_n^i V_i$ (d is the dimension of the associated vector fields).

Consider that the probability distribution of χ_n is $p(\chi_n)$. The additional information about χ_n is obtained from the measurement y_n as:

$$y_n = h(\chi_n) + v_n \tag{28}$$

where h is the observation function, and $v_n \sim \mathcal{N}(0, \mathrm{R}_n)$ denotes the white Gaussian noise. The UKF-M module used the gyro measurements and acceleration as inputs to update the random variable χ. The measurements of this standard 3D kinematic model were represented as:

$$y_n = \left\{\mu_n \in \mathbb{R}^3, a_{b_n} \in \mathbb{R}^3\right\} \tag{29}$$

where $\mu_n = \left(\omega_{x_n}, \omega_{y_n}, \omega_{z_n}\right)$ represents the gyroscope, and $a_{b_n} = \left(a_{x_n}, a_{y_n}, a_{z_n}\right)$ denotes the accelerometer. The UKF-M algorithm [34] updated the state and covariance by combining measurements y_n and system states χ_n.

In PINN UKFM, the output of the PINN module was utilized to filter the noise and minimize the norm errors. Pseudo-states $\hat{u}_\theta$ served as the calibrated IMU measurements for the filtering process. The states of the pseudo-states were represented as:

$$y_n = {y_n}' = \hat{u}_\theta = \left\{\mu_n \in \mathbb{R}^3, a_{b_n} \in \mathbb{R}^3\right\} \tag{30}$$

where ${y_n}' = \hat{u}_\theta$ represents the PINN module output as the pseudo-states.

Using the propagation function [34], PINN UKFM built the vehicle model. First, the gyro measurements were inputted to calculate the rotation matrix:

$$C_{n+1} = C_n \exp\left(\left(\mu_n - b_{g_n} + w_n^{(0:3)}\right) \times d_t\right) \tag{31}$$

where exp is the exponential map on the SO(3), and d_t is the integration step. In this vehicle model, $w_n^{(0:12)}$ represented the noise, and $w_n^{(0:3)}$, $w^{(3:6)}$, $w^{(6:9)}$, and $w^{(9:12)}$ were the submatrices of $w_n^{(0:12)}$. Next, the vehicle acceleration was inputted to calculate the calibrated vehicle acceleration:

$$a = C_n\left(a_{b_n} - b_{a_n} + w^{(3:6)}\right) + g \tag{32}$$

where $g = [0, 0, -9.82]$ represents the gravitational constant and a represents the calibrated vehicle acceleration. Based on the calibrated vehicle acceleration, the model updated the vehicle speed as:

$$v_{n+1} = v_n + ad_t \tag{33}$$

where $v_n = (v_{E_n}, v_{N_n}, v_{U_n})$ denotes the velocity vector. Based on the vehicle velocity vector, the model updated the vehicle position vector as:

$$\mathscr{P}_{n+1} = \mathscr{P}_n + v_n \times d_t + \frac{a \times {d_t}^2}{2} \tag{34}$$

where $\mathscr{P}_n = (E_n,\ N_n, U_n)$ is the vehicle coordinates in the navigation coordinates. Finally, the model uploaded the gyro and accelerometer biases as:

$$b_{a,n+1} = b_{a,n} + w^{(6:9)} \times d_t b_{g,n+1} = b_{g,n} + w^{(9:12)} \times d_t \tag{35}$$

where $b_{g,n}$ represents the gyro bias, and $b_{a,n}$ represents the accelerometer bias.

The probability distribution of χ and the propagation function remained unchanged; therefore, the posterior distribution $p(\chi_n|{y_n}')$ was calculated as $p(\chi_n|y_n)$. The pseudo-states ${y_n}'$ provided information about the sigma point ξ_n. First, the sigma points ξ_n were computed as:

$$\xi_{j_n} = \begin{cases} \mathrm{col}\left(\sqrt{(\lambda + \mathrm{d})\mathrm{P}_n}\right)_j, j = 1, \ldots, d \\ -\mathrm{col}\left(\sqrt{(\lambda + \mathrm{d})\mathrm{P}_n}\right)_j, j = d+1, \ldots, 2d \\ \lambda = \left(\alpha^2 - 1\right)d \end{cases} \tag{36}$$

where λ is the scale parameter [48]; α is a free parameter chosen by the practitioner [49] (α must be small); d is the dimension of the associated vector fields; P_n is the covariance matrix at timestep n; and col represents that the j column of the matrix is the weight associated with the j point. Second, these sigma points passed through the vehicle model to yield the set of transformed sigma point y_{j_n}:

$$y_{j_n} = \begin{cases} h(\boldsymbol{\varphi}(\hat{\chi}_n, 0)), j = 0 \\ h\left(\boldsymbol{\varphi}\left(\hat{\chi}_n, \xi_{j_n}\right)\right), j = 1, \ldots, 2d \end{cases} \tag{37}$$

where $h(\boldsymbol{\varphi}(\hat{\chi}_n, 0))$ is the unnoisy state model, h is the observation function (as in Equations (31–35)), and $\hat{\chi}_n$ is the prior mean estimate of the current state. Third, the mean and covariance of the transformed sigma points [30] were computed as:

$$\begin{gathered} \overline{y_n} = \varpi_m y_{0_n} + \textstyle\sum_{j=1}^{2d} \varpi_j y_{j_n} \\ \mathrm{P}_{y_n y_n} = \textstyle\sum_{j=0}^{2d} \varpi_j \left(y_{j_n} - \overline{y_n}\right)\left(y_{j_n} - \overline{y_n}\right)^{\mathrm{T}} + \mathrm{R}_n \\ \mathrm{P}_{\xi_n y_n} = \textstyle\sum_{j=1}^{2d} \varpi_j \xi_{j_n} \left(y_{j_n} - \overline{y_n}\right) \\ \varpi_m = \frac{\lambda}{\lambda + d},\ \varpi_j = \frac{1/2}{\lambda + d} \end{gathered} \tag{38}$$

where $\overline{y_n}$ represents the mean of the transformed sigma points; $\mathrm{P}_{y_n y_n}$ is the covariance of the transformed sigma points; $\mathrm{P}_{\xi_n y_n}$ is the cross-covariance of the transformed sigma points; ϖ_m and ϖ_j are weights; and R_n is the covariance matrix of white Gaussian noise. Next, PINN UKFM employed the Kalman updated equation to update the state and covariance as:

$$\begin{gathered} \mathrm{K}_n = \mathrm{P}_{\xi_n y_n} \mathrm{P}_{y_n y_n}^{-1} \\ \hat{\chi}_n^{+} = \boldsymbol{\varphi}\left(\hat{\chi}_n, \mathrm{K}_n\left({y_n}' - \overline{y_n}\right)\right) \\ {\mathrm{P}_n}^{+} = \mathrm{P}_n - \mathrm{K}_n \mathrm{P}_{y_n y_n} {\mathrm{K}_n}^{\mathrm{T}} \end{gathered} \tag{39}$$

where ${y_n}'$ is the PINN module output (pseudo-states), K_n is the gain matrix, $\hat{\chi}_n^{+}$ is the posterior mean estimate state, P_n is the prior estimate covariance matrix of the current state, and ${\mathrm{P}_n}^{+}$ is the posterior estimate covariance matrix.

The unscented transformation [50] was employed to approximate the posterior $p(\xi_n|{y_n}')$ for ξ_n as:

$$\begin{gathered} p(\xi_n|{y_n}') \sim \mathcal{N}\left(\overline{\xi_n}, {\mathrm{P}_n}^{+}\right) \\ \overline{\xi_n} = \mathrm{K}_n\left({y_n}' - \overline{y_n}\right) \end{gathered} \tag{40}$$

where $\overline{\xi_n}$ represents the noise-free mean.

The unscented approximation to the posterior $p(\xi_n|y_n{}')$ was thus the distribution of a Gaussian $\overline{\xi_n} + \xi_n{}^+$ with $\xi_n{}^+ \sim \mathcal{N}(0, \mathrm{P}_n{}^+)$ [34]. Then, PINN UKFM approximated the posterior distribution $p(\chi_n|y_n{}')$ as:

$$\begin{gathered} \chi_n \approx \boldsymbol{\varphi}(\hat{\chi}_n^+, \xi_n{}^+),\ \xi_n{}^+ \sim \mathcal{N}(0, \mathrm{P}_n{}^+) \\ \hat{\chi}_n^+ = \boldsymbol{\varphi}(\hat{\chi}_n, \overline{\xi_n}) \\ \chi_n \approx \boldsymbol{\varphi}(\boldsymbol{\varphi}(\hat{\chi}_n, \overline{\xi_n}), \xi_n{}^+) \end{gathered} \tag{41}$$

where $\xi_n{}^+$ represents the posterior noise. The posterior distribution $p(\chi_n|y_n{}')$ boiled down to a Bayesian estimation problem [47] that incorporated the information from the PINN module.

In this paper, we focused on describing how PINN UKFM updated the state estimation $\hat{\chi}_n^+$ and covariance matrix $\mathrm{P}_n{}^+$ when pseudo-states $y_n{}'$ arrived. Additionally, the UKF-M algorithm could propagate the state without sensor measurement [34], which was implemented in our program.

4. Experimental Results

4.1. Vehicle Platform

A vehicle platform was built to validate the proposed algorithm. The hardware configuration is shown in Figure 4. The sensors included GNSS, IMU, MSW, WFT, S-Motion, and a camera. D[WE-43A-USB and Dewesoft SIRIUS acquisition systems were used. The DEWE-43A-USB system obtained the S-Motion and MSW signals through high-speed CAN channels, while the Dewesoft SIRIUS system collected the WFT signals through high-speed CAN channels. A computer was used for all data acquisition.

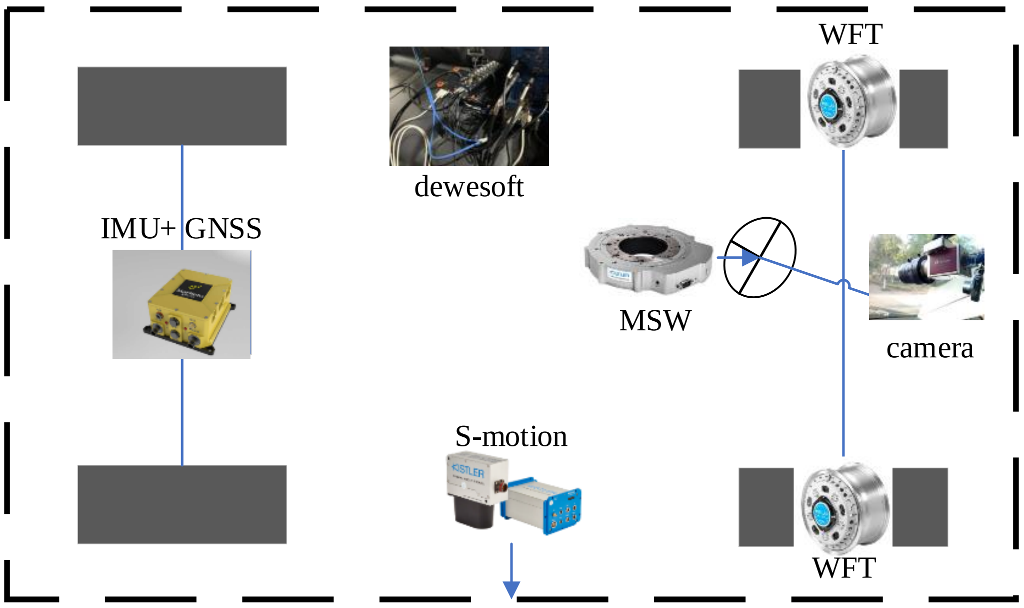

Figure 4. Hardware configuration.

First, the Mako G-192B monocular camera [51], produced by the Allied Vision Company, captured image data of the environment. Then, the starNeto XW-GI [52] provided the IMU and GNSS data, using differential methods to measure both position and attitude. $GPFPD was selected as the starNeto communication protocol to acquire signals for $[E, N, U, \varphi, \Theta, \psi, v_e, v_n, v_z]$. In the third step, the S-Motion system, produced by Kistler, provided signals for $\left[v_x, a_x, v_y, a_y, a_z, \omega_x, \omega_y, \omega_z\right]$. In the fourth step, the MSW by Kistler was used to measure the steering wheel angle, speed, and torque. Finally, the WFT by Kistler was used to measure the wheel forces and moments under dynamic conditions for the left and right front wheels. The WFT provided signals for $\left[F_x, M_x, F_y, M_y, F_z, M_z\right]$ in the front wheels and three axes of WFT, similar to the vehicle body coordinates. The hardware implementation in the test vehicle [52–56] is shown in Figure 5.

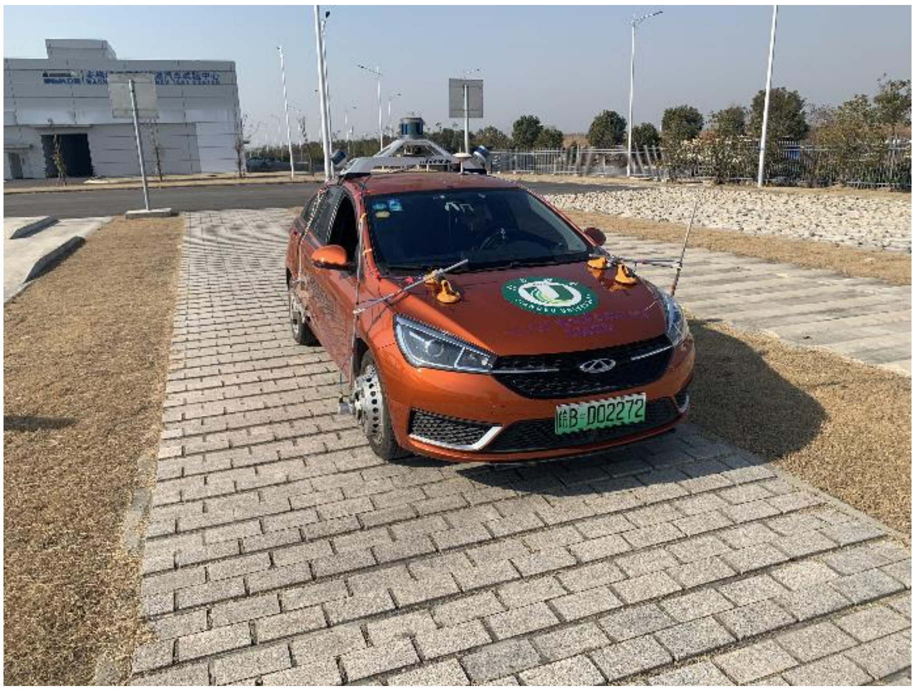
Figure 5. The test vehicle.

The test was conducted on cement pavement at Jiangsu University, and the vehicle trajectory is shown in Figure 6. The weather during the test was rainy, which reduced the tire–road friction coefficient and increased the nonlinearity of the vehicle's dynamic relationships. A total of 85% of the data were used for training and to verify the PINN module, while the remaining 15% were reserved for testing. The dataset included scenarios such as an overpass (high speed), a school (low speed), a slope, traffic lights, and vehicle turning. The road scene could be changed by altering the dataset splits. In the experiment, the last 15% of the dataset was used to evaluate the performance of PINN UKFM.

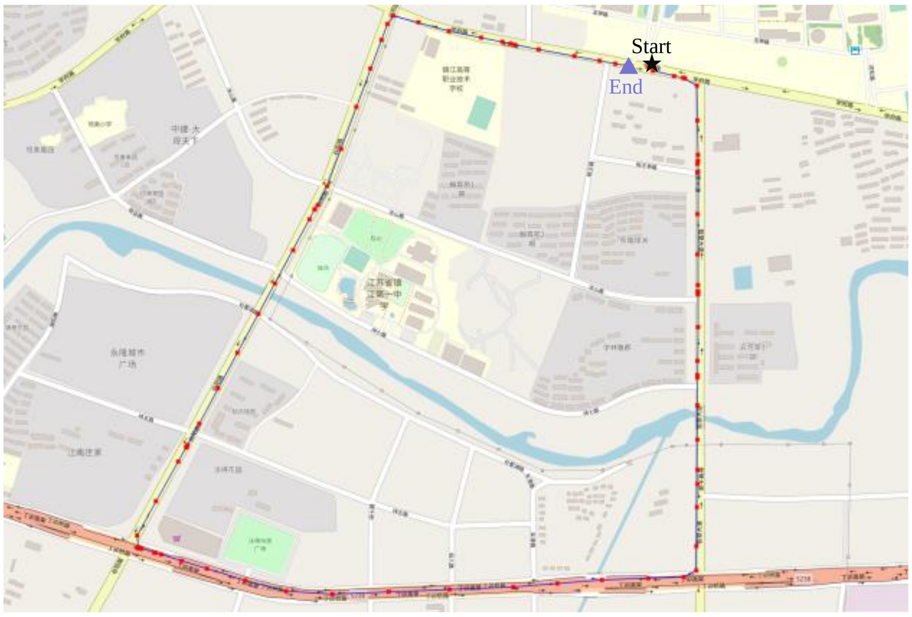

Figure 6. The trajectory of the experiment.

4.2. Parameter Settings and Training of All Comparative Methods

PINN: The structure of the PINN was chosen as {22, 32, 32×5, 64, 128, 128, 64, 32, 6}. For easy calculation of v_z, we subtracted the gravity vector from the acceleration a_z during data processing [34]. The layer {22, 32} was the encoder layer, and the layers {32, 32×5, 64, 128, 128, 64, 32} were the temporal interaction layers. The layer {32, 6} was the decoder layer. The prediction states of the PINN module were the IMU calibration values.

subPINN: To address the influence of loss backward during training, a single-output scheme of the PINN was implemented. The vehicle dynamics/kinematic models were used to build the data-driven model $\left[a_x, a_y, \omega_z\right]$. This subPINN had the same structure as the PINN and a single output. Specifically, for training $\left[a_z, \omega_x, \omega_y\right]$, the structure {1, 2, 32, 32, 64} was used to reduce the computational complexity and improve accuracy. The inputs of $\left[a_z, \omega_x, \omega_y\right]$ were $[a_z, v_z]$, $[\omega_x, \varphi]$, and $[\omega_y, \Theta]$, respectively. Different from the UTM coordinates, the attitudes did not differ greatly, so they were directly inputted here.

MLP [24,25,57]: The multilayer perceptron (MLP) is a commonly used approach for building data-driven vehicle dynamics models. The structure of the MLP used in this study was inspired by [24] and was chosen as {22, 32, 32×5, 64, 128, 128, 64, 32, 6}. The ReLU activation function was applied between each layer. Different from the approach used in [24], we used the layers {32, 32×5} as the temporal interaction layers, which is a common approach in data-driven dynamics modeling [25].

LSTM [24,57,58]: Long Short-Term Memory (LSTM) is another classic method used to build data-driven dynamics models, which is employed for predicting the acceleration and angular velocity of vehicles [57,58]. LSTM is known for its ability to capture temporal dependencies in the data. In this study, LSTM was implemented with a general transformation structure as {22, 32, 128, 32, 6}, where {22, 32} and {32, 6} represented the MLP layers. The exponential linear unit (ELU) [35] function was applied after the first layer, and {32, 32, 128} represented the two LSTM layers.

Physical models [23,45]: The yaw rate formulas based on longitudinal/lateral dynamics and kinematic models were used in this paper. The longitudinal dynamics parameters were calculated using the least squares method [23]. The lateral dynamics parameters were computed based on the actual vehicle parameters (l_f, l_r, m) and the relationships between states ($\mathrm{C} = \frac{F_y}{a_y}$). We assumed the yaw rate was an LTI state. Therefore, a kinematic model [45] was used to calculate the yaw rate.

The wheel force $[F_x, F_y]$ measured by WFT could be utilized to calculate the acceleration through algebraic equations. Additionally, we used the linearly constrained least squares method to combine these dynamics as:

$$\begin{aligned} \mathrm{m}a_{x_n} &= \left(\mathscr{K}_1 \times M_{y_{l,n}} + \mathscr{K}_2 \times M_{y_{r,n}} + \mathscr{K}_3 \mathrm{mg} - \mathscr{K}_4 v_{x_n}^2\right) + \mathscr{K}_5 F_{x_{fl,n}} + \mathscr{K}_6 F_{x_{fr,n}} + \mathscr{K}_7 \\ \mathrm{m}a_{y_n} &= \mathscr{K}_8\left(\frac{-4\mathrm{C}v_{y_n} - 2\mathrm{C}\left(l_f - l_r\right)\omega_{z_n}}{v_{x_n}\mathrm{m}} - \mathrm{m}v_{x_n}\omega_{z_n} + \frac{2\mathrm{C}\delta_n}{\mathrm{m}}\right) + \mathscr{K}_9 F_{y_{fl,n}} + \mathscr{K}_{10} F_{y_{fr,n}} + \mathscr{K}_{11} \end{aligned} \tag{42}$$

where $\mathscr{K}_1 \sim \mathscr{K}_6, \mathscr{K}_8 \sim \mathscr{K}_{10}$ represent the parameters of the least squares method, and $\mathscr{K}_7$ and $\mathscr{K}_{11}$ represent the coefficients of the least squares method.

The resulting states of the physical models are illustrated in Figure 7. In the experiment, we not only compared the prediction states with the sensor signals but also compared the integration states $\left[v_x, v_y, v_z, \varphi, \Theta, \psi\right]$ with the integrated sensor signals, which could be expressed as:

$$\mathscr{X}_{\mathrm{t}} = \mathscr{X}_0 + \int_{i=0}^{\mathrm{t}} u_i dt \tag{43}$$

where $\mathscr{X}_0$ is initial state, t is the timestamp, and $u_i = \left[a_x, a_y, a_z, \omega_x, \omega_y, \omega_z\right]$ are the states at timestamp i.

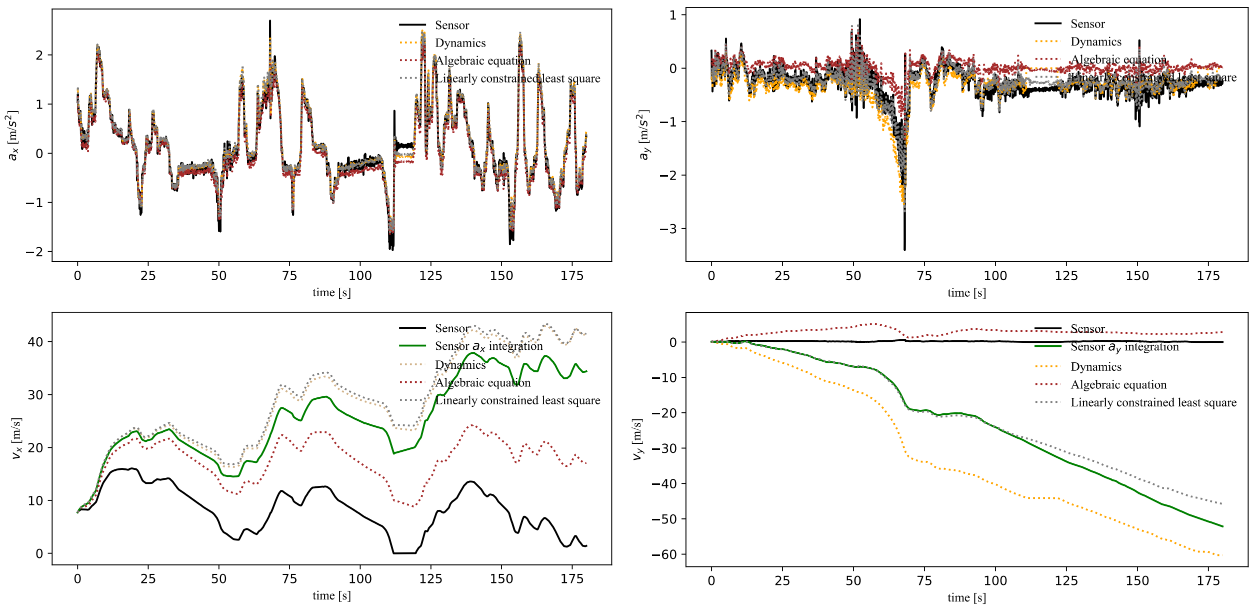

Figure 7. The results of physical models.

As shown in Figure 7, the "Algebraic equation" represented the calculation results of Equations (16) and (18). The "Linearly constrained least square" represented the calculation results of Equation (42). By combining the forces and dynamics, the output of this equation provided a better fit to the IMU measurements. This method could fit multiple sensors to improve the accuracy of the dynamics, when the IMU measurements were accurate. However, due to the IMU drift, the integration error of this model was relatively large. The "Dynamics" represented the longitudinal/lateral dynamics model based on Equations (15) and (17), which is commonly used for vehicle control. In this paper, "Linearly constrained least square" and "Dynamics" were established based on the IMU measurement data. Therefore, the results of these models were close to the IMU measurements but also had large integration errors.

To maintain visual clarity, we only used the vehicle dynamics ("Dynamics") and kinematic models for comparison. MLP and LSTM also utilized the PyTorch deep learning framework. We employed the Adam optimizer with a 0.001 learning rate to train the network. The models were trained using the Nvidia GTX 3080Ti GPU. All models, including PINN, subPINN, MLP, and LSTM, were trained using the same training dataset. However, there was a difference in the loss calculation method. Different from the loss calculation method established by PINN, MLP and LSTM used the IMU measurements to build data-driven vehicle models [25,57,58]. These methods commonly assumed that the ground truth states of the vehicle were accessible. In this paper, we used highly accurate S-motion data as a surrogate for the actual vehicle state for MLP and LSTM training.

4.3. Validation of the PINN Module

In this subsection, we compare the PINN module with the MLP, LSTM, and vehicle dynamics/kinematic models and vehicle measurements. These data-driven models were trained using the dataset described in Section 4.1. The root mean square error (RMSE) [24,59] was used to evaluate the performance of the models, which was represented as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathcal{S}}\sum_{i=1}^{\mathcal{S}}(\varkappa_i - \hat{\varkappa}_i)^2} \tag{44}$$

where $\mathcal{S}$ = 1800 is the predicted steps, $\varkappa_i = [u_i, \mathcal{Z}_i]$ represents the i-th value of the predicted results, and $\hat{\varkappa}_i$ represents the i-th value of the reference results (sensor measurements).

Table 2 presents the RMSEs of the predicted states using different methods. The results demonstrated that except for longitudinal/lateral acceleration, the PINN-based approaches (PINN and subPINN) estimated the vehicle state more accurately. The inferior prediction performance on longitudinal/lateral acceleration was attributed to sensor drift. By integrating the acceleration to obtain velocity, it was shown that the PINN-based methods could effectively take the influence of other sensors and realize IMU calibration.

Table 2. The RMSEs of the predicted states using different methods.

Model	a_x	a_y	a_z	ω_x	ω_y	ω_z
MLP	0.1117	0.1524	0.2729	0.0164	0.0171	0.0174
LSTM	0.1167	0.1444	0.2626	0.0171	0.0175	0.0167
PINN	0.2733	0.3904	0.2806	0.0110	0.0084	0.0058
subPINN	0.2301	0.3606	0.2167	0.0076	0.0049	0.0036
Model	v_x	v_y	v_z	φ	Θ	ψ
MLP	20.2365	25.2757	2.7479	0.1027	0.1865	0.3334
LSTM	19.6955	24.8413	0.5609	0.0613	0.1391	0.3856
PINN	1.3007	4.6707	0.5609	0.0806	0.0429	0.0475
subPINN	1.8171	7.3559	0.4169	0.0312	0.0113	0.0217

As shown in Figure 8, the estimation results of $\left[a_x, v_x, a_y, v_y\right]$ were obtained. "Sensor a_x integration" was the ODE states inferred from the original signals using Equation (43) and is shown in green. The PINN model incorporated both longitudinal/lateral vehicle dynamics and LTI-based vehicle displacement, which yielded higher-precision results than the subPINN model. Compared with the MLP and LSTM models, the PINN model had better prediction accuracy of the integrated states. This was due to the PINN model being able to incorporate more dynamics knowledge into the data-driven model.

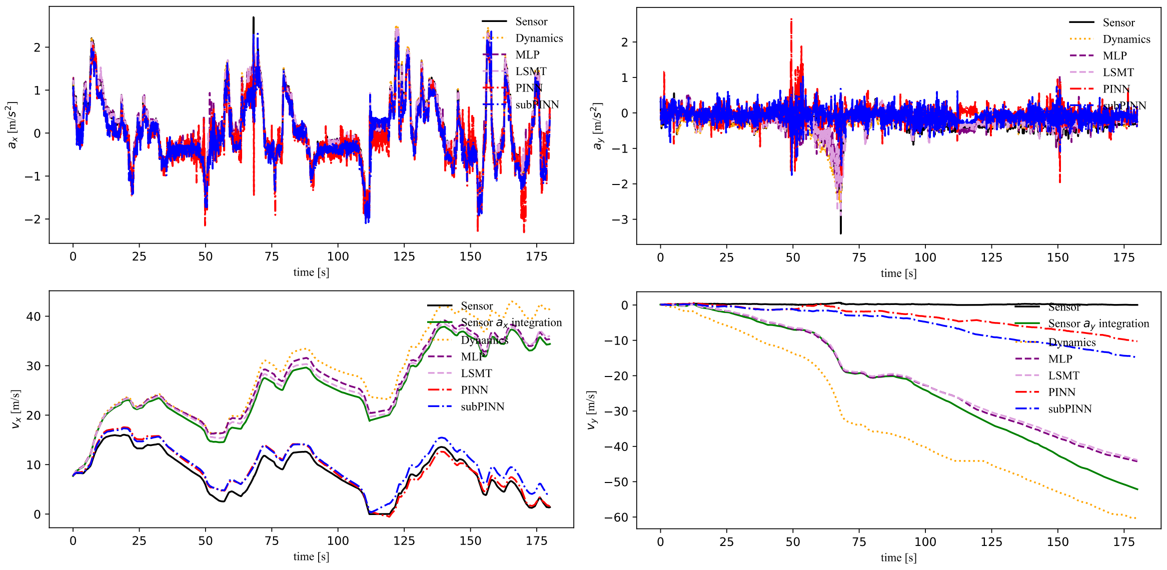

Figure 8. The prediction results of states $\left[a_x, v_x, a_y, v_y\right]$ and sensor measurements.

The estimation results of states $[a_z, v_z, \omega_x, \varphi, \omega_y, \Theta, \omega_z, \psi]$ were obtained as shown in Figure 9. The experiment of $[a_z, \omega_x, \omega_y]$ aimed to reduce the noise in acceleration and angular velocity by combining the PINN module with the linear ODEs. As the noise impact on the S-Motion sensor increased over time, the subPINN model demonstrated superior accuracy in estimating the state variables while maintaining the sensor data characteristics and reducing noise. Both the PINN and subPINN models used the vehicle kinematic model in modeling ω_x. The subPINN model converged more readily in yaw rate prediction than the other models. Compared to the MLP and LSTM models, the subPINN model had better prediction accuracy in each state.

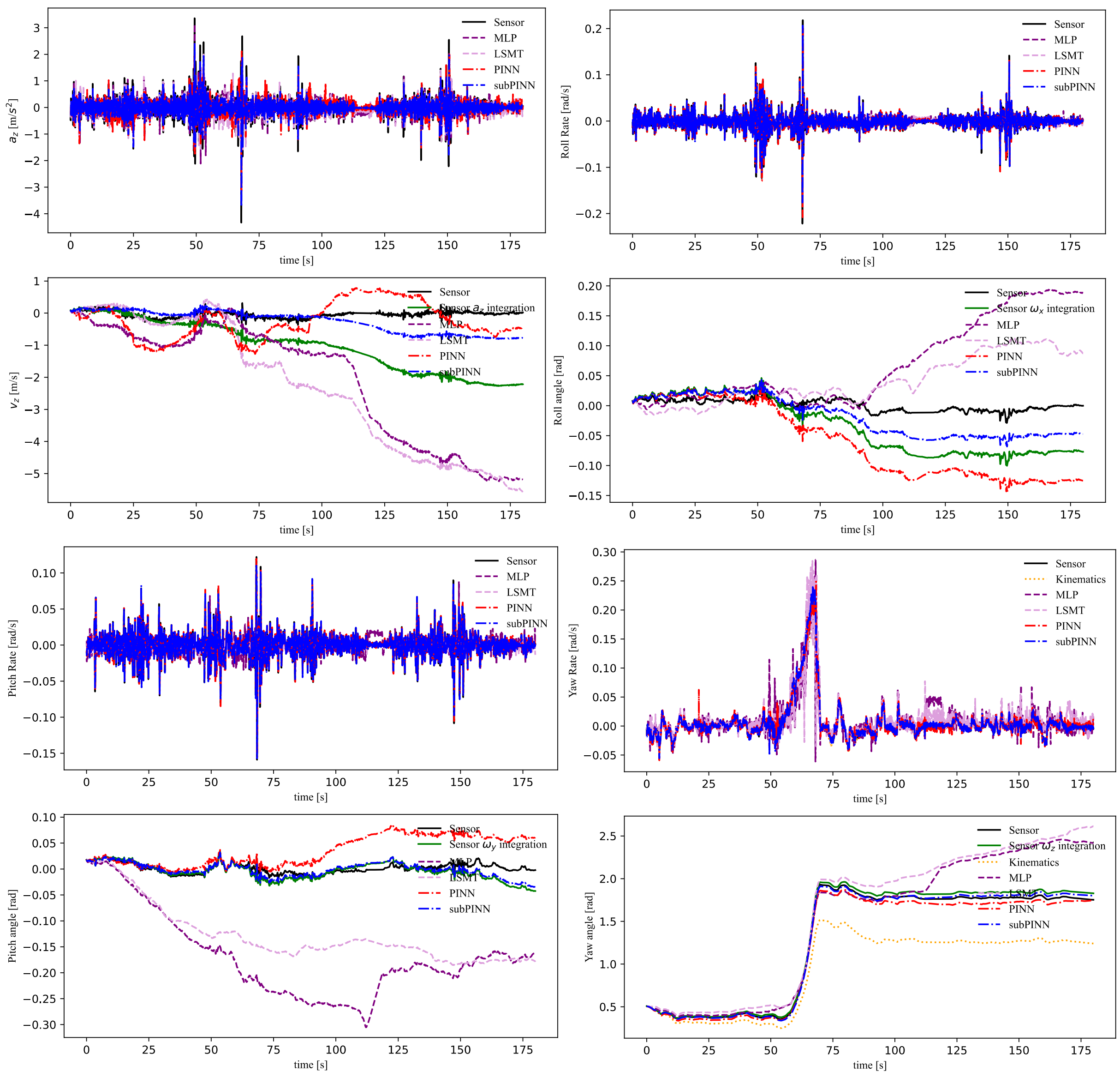

Figure 9. The prediction results of states $[a_z, v_z, \omega_x, \varphi, \omega_y, \Theta, \omega_z, \psi]$ and sensor measurements.

The subPINN model estimated a single vehicle state, which simplified the training process and enabled easier convergence during training. In contrast, the training process of the PINN model was more complex and it was difficult to achieve optimal results. The subPINN model offered advantages in terms of training efficiency and high accuracy results. This gave the subPINN model a certain attractiveness for engineering implementations. However, the subPINN model could not establish connections between multiple states like the PINN model in order to achieve more comprehensive modeling. Therefore, the PINN model performed better in terms of longitudinal/lateral dynamics accuracy.

4.4. Validation of PINN UFM

The experimental results presented above demonstrated the advantages of the PINN module, but some issues in the comparison still remained. The vehicle states were not connected through a vehicle-based model during the comparison, which failed to reflect the impact of the estimation states on the vehicle dynamics. Compared to other vehicle states, the cm-level GNSS positioning could better reflect the effectiveness of the presented estimation states.

In this subsection, the UKF-M module based on the output of the PINN module is demonstrated. This module reduced state noise through the Gaussian noise hypothesis. Figure 10 shows the trajectories of PINN UKFM and sensor inputs. The vectors $[E, N, h]$ were entered at 20 Hz. The covariance matrices of the point were iterated quickly, which could not encapsulate the performance improvement of PINN UKFM.

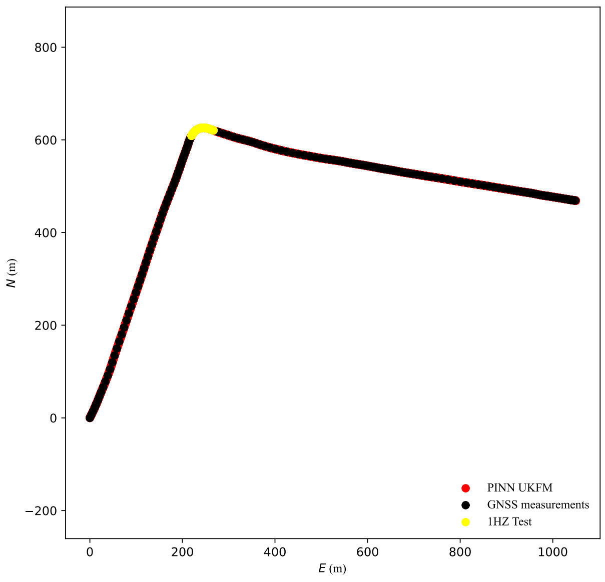

Figure 10. Compared trajectories at 20 Hz GNSS position.

Therefore, the frequency of the trajectory coordinates was reduced to 1 Hz through downsampling. As illustrated in Figure 11, the results obtained using PINN UKFM were superior to those obtained using the conventional UKF-M algorithm. Without coordinate input, the vehicle coordinates were updated in the vehicle model using the UKF-M module. The PINN module could reduce the influence of vehicle sensor drift and provided a reliable state estimation result that was closer to the true state of the vehicle. Therefore, the updated vehicle position of PINN UKFM was also closer to the true vehicle state.

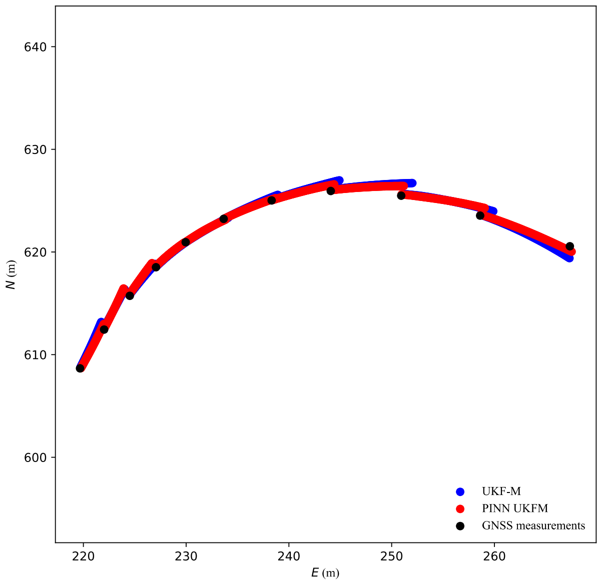

Figure 11. Compared trajectories at 1 Hz GNSS position.

Additionally, the UKF-M module could estimate the vehicle speed, which included $[v_\mathrm{e}, v_\mathrm{n}, v_\mathrm{u}]$. However, the $[v_\mathrm{e}, v_\mathrm{n}, v_\mathrm{u}]$ were not inputted into the PINN module. Next, the modeling results were compared to the measurement states to verify PINN UKFM correctness. As shown in Figure 12, compared to the UKF-M algorithm, PINN UKFM could better estimate v_u.

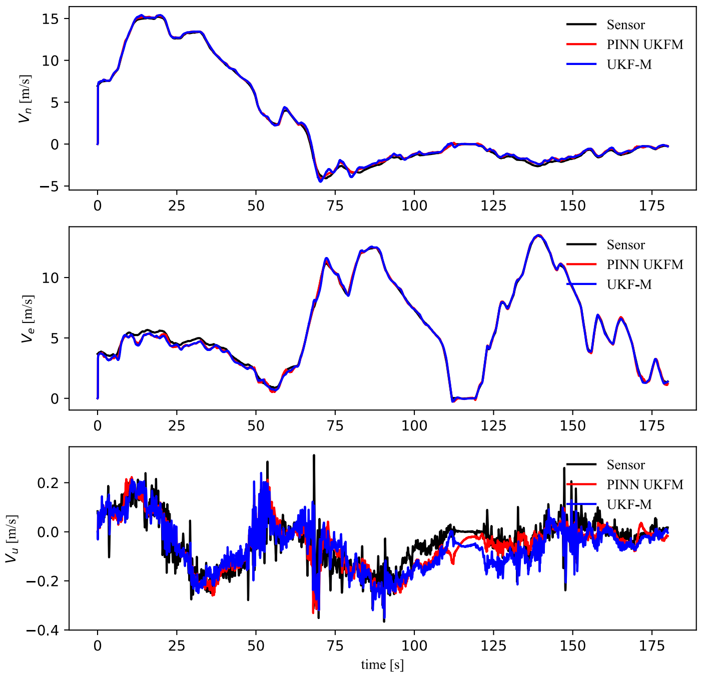

Figure 12. Compared 3D velocities at 20HZ GNSS position.

Considering that the cm-level GNSS positioning has a great impact on the covariance matrix, the velocity at the 1 Hz GNSS position was also compared, as shown in Figure 13. The PINN UKFM results were significantly better than the UKF-M results.

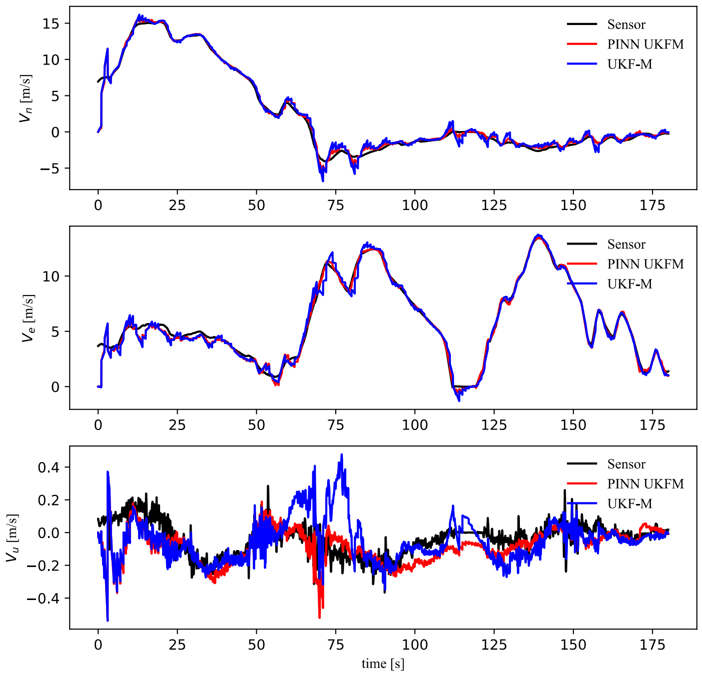

Figure 13. Compared 3D velocities at 1HZ GNSS position.

Evidently, PINN UKFM exhibited remarkable performance in vehicle state estimation, although there was still potential for further improvement. A possible enhancement lies in leveraging the signals from the CAN bus to replace the WFT and MSW sensors. The CAN bus grants access to a wealth of vehicle sensor data, including valuable information such as the steering angle, throttle opening, and brake pressure. By incorporating these signals as inputs to the model, a dynamic vehicle model could be constructed, enabling more accurate state estimation. Additionally, the S-Motion sensor can also be replaced by a cheaper IMU. Using more universal CAN bus signals and IMUs, the model could be implemented on more vehicle platforms without being limited by specific sensors.

5. Conclusions

In this study, a novel method for vehicle state estimation combining PINN and UKF-M has been proposed. The sensor module collects various sensor signals and feeds them into the PINN module. The PINN module automatically calibrates the IMU by utilizing the ODE relationships that exist between multiple sensors. The PINN-based model mitigates sensor errors and facilitates sensor fusion, leveraging the LTI hypothesis for loss calculation in order to reduce noise and bias in the original sensor data. It exhibits clear advantages in multi-sensor fusion compared to existing data-driven vehicle dynamics models. The resulting pseudo-states are then used as inputs to the UKF-M module to model the vehicle trajectory. The experimental results confirmed its effectiveness, with the PINN module successfully eliminating IMU drift and the PINN UKFM trajectory exhibiting less deviation from the cm-level GNSS positioning than the UKF-M trajectory. In the future, we aim to explore more cost-effective solutions to implement the PINN UKFM method. Additionally, the authors intend to use PINN UKFM in OCCS control.

Author Contributions: Conceptualization, C.T., Y.C. and X.S.; Methodology, C.T.; Software, C.T.; Resources, Y.C.; Writing—original draft, C.T.; Writing—review & editing, Y.C.; Supervision, Y.C., H.W. and L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Key Research and Development Program of China (2022YFB2503302), National Natural Science Foundation of China (52225212, U20A20333, U20A20331, 52072160, 52272418, U22A20100), Key Project for the Development of Strategic Emerging Industries of Jiangsu Province (BE2020083-3, BE2020083-2).

Data Availability Statement: The dataset from our experiments and model can be downloaded in Google drive. Moreover, we have developed a Jupyter notebook (script/PINN_UKFM_TEST.ipynb) that enables online access to all experimental results. https://drive.google.com/drive/folders/1iKkIuy6L8LUOVyypgxCLsdL_lAH1mCjG, accessed on 1 June 2023.

Acknowledgments: We would like to express our gratitude to Yubo Lian and Yilin Zhong from BYD Company for their valuable assistance in the installation of sensors on the autonomous vehicle. Their support has greatly contributed to our research project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guerrero-Ibáñez, J.; Zeadally, S.; Contreras-Castillo, J. Sensor Technologies for Intelligent Transportation Systems. *Sensors* **2018**, *18*, 1212. [CrossRef]
2. Kissai, M.; Monsuez, B.; Tapus, A. Review of integrated vehicle dynamics control architectures. In Proceedings of the 2017 European Conference on Mobile Robots (ECMR), Paris, France, 6–8 September 2017; pp. 1–8. [CrossRef]
3. Kissai, M. Optimal Coordination of Chassis Systems for Vehicle Motion Control. Ph.D. Thesis, Université Paris-Saclay (ComUE), Orsay, France, 2019.
4. Zhang, L.; Zhang, Z.; Wang, Z.; Deng, J.; Dorrell, D. Chassis coordinated control for full X-by-wire vehicles-A review. *Chin. J. Mech. Eng.* **2021**, *34*, 42. [CrossRef]
5. Xia, X.; Xiong, L.; Lu, Y.; Gao, L.; Yu, Z. Vehicle sideslip angle estimation by fusing inertial measurement unit and global navigation satellite system with heading alignment. *Mech. Syst. Signal Process.* **2021**, *150*, 107290. [CrossRef]
6. Melzi, S.; Sabbioni, E. On the vehicle sideslip angle estimation through neural networks: Numerical and experimental results. *Mech. Syst. Signal Process.* **2011**, *25*, 2005–2019. [CrossRef]

7. Yin, Y.; Zhang, J.; Guo, M.; Ning, X.; Wang, Y.; Lu, J. Sensor Fusion of GNSS and IMU Data for Robust Localization via Smoothed Error State Kalman Filter. *Sensors* **2023**, *23*, 3676. [CrossRef]
8. Laftchiev, E.I.; Lagoa, C.M.; Brennan, S.N. Vehicle localization using in-vehicle pitch data and dynamical models. *IEEE Trans. Intell. Transp. Syst.* **2014**, *16*, 206–220. [CrossRef]
9. Xia, X.; Xiong, L.; Huang, Y.; Lu, Y.; Gao, L.; Xu, N. Estimation on IMU yaw misalignment by fusing information of automotive onboard sensors. *Mech. Syst. Signal Process.* **2022**, *162*, 107993. [CrossRef]
10. Song, R.; Fang, Y. Vehicle state estimation for INS/GPS aided by sensors fusion and SCKF-based algorithm. *Mech. Syst. Signal Process.* **2021**, *150*, 107315. [CrossRef]
11. Wang, W.; Liu, Z.; Xie, R. Quadratic extended Kalman filter approach for GPS/INS integration. *Aerosp. Sci. Technol.* **2006**, *10*, 709–713. [CrossRef]
12. Zhang, W.; Wang, Z.; Zou, C.; Drugge, L.; Nybacka, M. Advanced vehicle state monitoring: Evaluating moving horizon estimators and unscented Kalman filter. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5430–5442. [CrossRef]
13. Hauberg, S.; Lauze, F.; Pedersen, K.S. Unscented Kalman filtering on Riemannian manifolds. *J. Math. Imaging Vis.* **2013**, *46*, 103–120. [CrossRef]
14. Du, S.; Huang, Y.; Lin, B.; Qian, J.; Zhang, Y. A lie group manifold-based nonlinear estimation algorithm and its application to low-accuracy SINS/GNSS integrated navigation. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–27. [CrossRef]
15. Chen, T.; Cai, Y.; Chen, L.; Xu, X.; Jiang, H.; Sun, X. Design of vehicle running states-fused estimation strategy using Kalman filters and tire force compensation method. *IEEE Access* **2019**, *7*, 87273–87287. [CrossRef]
16. Park, G.; Choi, S.B.; Hyun, D.; Lee, J. Integrated observer approach using in-vehicle sensors and GPS for vehicle state estimation. *Mechatronics* **2018**, *50*, 134–147. [CrossRef]
17. Šabanovič, E.; Kojis, P.; Šukevičius, Š.; Shyrokau, B.; Ivanov, V.; Dhaens, M.; Skrickij, V. Feasibility of a Neural Network-Based Virtual Sensor for Vehicle Unsprung Mass Relative Velocity Estimation. *Sensors* **2021**, *21*, 7139. [CrossRef]
18. Kim, D.; Min, K.; Kim, H.; Huh, K. Vehicle sideslip angle estimation using deep ensemble-based adaptive Kalman filter. *Mech. Syst. Signal Process.* **2020**, *144*, 106862. [CrossRef]
19. Kim, D.; Kim, G.; Choi, S.; Huh, K. An integrated deep ensemble-unscented Kalman filter for sideslip angle estimation with sensor filtering network. *IEEE Access* **2021**, *9*, 149681–149689. [CrossRef]
20. Vargas-Meléndez, L.; Boada, B.L.; Boada, M.J.L.; Gauchía, A.; Díaz, V. A Sensor Fusion Method Based on an Integrated Neural Network and Kalman Filter for Vehicle Roll Angle Estimation. *Sensors* **2016**, *16*, 1400. [CrossRef]
21. Soriano, M.A.; Khan, F.; Ahmad, R. Two-axis accelerometer calibration and nonlinear correction using neural networks: Design, optimization, and experimental evaluation. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 6787–6794. [CrossRef]
22. Boada, B.L.; Boada, M.J.L.; Diaz, V. Vehicle sideslip angle measurement based on sensor data fusion using an integrated ANFIS and an Unscented Kalman Filter algorithm. *Mech. Syst. Signal Process.* **2016**, *72*, 832–845. [CrossRef]
23. Vicente, B.A.H.; James, S.S.; Anderson, S.R. Linear system identification versus physical modeling of lateral–longitudinal vehicle dynamics. *IEEE Trans. Control. Syst. Technol.* **2020**, *29*, 1380–1387. [CrossRef]
24. Xiao, Y.; Zhang, X.; Xu, X.; Liu, X.; Liu, J. Deep neural networks with Koopman operators for modeling and control of autonomous vehicles. *IEEE Trans. Intell. Veh.* **2023**, *8*, 135–146. [CrossRef]
25. Spielberg, N.A.; Brown, M.; Kapania, N.R.; Kegelman, J.C.; Gerdes, J.C. Neural network vehicle models for high-performance automated driving. *Sci. Robot.* **2019**, *4*, eaaw1975. [CrossRef] [PubMed]
26. Xiao, Z.; Xiao, D.; Havyarimana, V.; Jiang, H.; Liu, D.; Wang, D.; Zeng, F. Toward accurate vehicle state estimation under non-Gaussian noises. *IEEE Internet Things J.* **2019**, *6*, 10652–10664. [CrossRef]
27. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
28. Xu, P.-F.; Han, C.-B.; Cheng, H.-X.; Cheng, C.; Ge, T. A Physics-Informed Neural Network for the Prediction of Unmanned Surface Vehicle Dynamics. *J. Mar. Sci. Eng.* **2022**, *10*, 148. [CrossRef]
29. Franklin, T.S.; Souza, L.S.; Fontes, R.M.; Martins, M.A. A Physics-Informed Neural Networks (PINN) oriented approach to flowmetering in oil wells: An ESP lifted oil well system as a case study. *Digit. Chem. Eng.* **2022**, *5*, 100056. [CrossRef]
30. Wong, J.C.; Chiu, P.H.; Ooi, C.C.; Da, M.H. Robustness of Physics-Informed Neural Networks to Noise in Sensor Data. *arXiv* **2022**, arXiv:2211.12042.
31. Arnold, F.; King, R. State–space modeling for control based on physics-informed neural networks. *Eng. Appl. Artif. Intell.* **2021**, *101*, 104195. [CrossRef]
32. Alatise, M.B.; Hancke, G.P. Pose Estimation of a Mobile Robot Based on Fusion of IMU Data and Vision Data Using an Extended Kalman Filter. *Sensors* **2017**, *17*, 2164. [CrossRef]
33. Gräber, T.; Lupberger, S.; Unterreiner, M.; Schramm, D. A hybrid approach to side-slip angle estimation with recurrent neural networks and kinematic vehicle models. *IEEE Trans. Intell. Veh.* **2018**, *4*, 39–47. [CrossRef]
34. Brossard, M.; Barrau, A.; Bonnabel, S. A code for unscented Kalman filtering on manifolds (UKF-M). In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 5701–5708.
35. Zhang, W.; Al Kobaisi, M. On the Monotonicity and Positivity of Physics-Informed Neural Networks for Highly Anisotropic Diffusion Equations. *Energies* **2022**, *15*, 6823. [CrossRef]

36. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1492–1500.
37. Rai, R.; Sahu, C.K. Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus. *IEEE Access* **2020**, *8*, 71050–71073. [CrossRef]
38. Fioretto, F.; Van Hentenryck, P.; Mak, T.W.; Tran, C.; Baldo, F.; Lombardi, M. Lagrangian duality for constrained deep learning. In Proceedings of the Applied Data Science and Demo Track: European Conference, ECML PKDD 2020, Part, V.. Ghent, Belgium, 14–18 September 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 118–135.
39. Bajaj, C.; McLennan, L.; Andeen, T.; Roy, A. Recipes for when Physics Fails: Recovering Robust Learning of Physics Informed Neural Networks. *Mach. Learn. Sci. Technol.* **2023**, *4*, 015013. [CrossRef]
40. Liu, D.; Wang, Y. A Dual-Dimer method for training physics-constrained neural networks with minimax architecture. *Neural Netw.* **2021**, *136*, 112–125. [CrossRef]
41. Chen, T.Q.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D.K. Neural ordinary differential equations. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Munich, Germany, 8–14 September 2018; pp. 6571–6583.
42. Yuan, L.; Ni, Y.Q.; Deng, X.Y.; Hao, S. A-PINN: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. *J. Comput. Phys.* **2022**, *462*, 111260. [CrossRef]
43. Cuomo, S.; Di Cola, V.S.; Giampaolo, F.; Rozza, G.; Raissi, M.; Piccialli, F. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *J. Sci. Comput.* **2022**, *92*, 88. [CrossRef]
44. Deo, N.; Trivedi, M.M. Convolutional social pooling for vehicle trajectory prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 1468–1476.
45. Kong, J.; Pfeiffer, M.; Schildbach, G.; Borrelli, F. Kinematic and dynamic vehicle models for autonomous driving control design. In Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, Republic of Korea, 28 June–1 July 2015; pp. 1094–1099.
46. Gao, Z.; Wang, J.; Wang, D. Dynamic modeling and steering performance analysis of active front steering system. *Procedia Eng.* **2011**, *15*, 1030–1035. [CrossRef]
47. Barrau, A.; Bonnabel, S. Intrinsic filtering on Lie groups with applications to attitude estimation. *IEEE Trans. Autom. Control* **2014**, *60*, 436–449. [CrossRef]
48. Barfoot, T.D.; Furgale, P.T. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Trans. Robot.* **2014**, *30*, 679–693. [CrossRef]
49. Brossard, M.; Bonnabel, S.; Condomines, J.P. Unscented Kalman filtering on Lie groups. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 2485–2491.
50. Julier, S.J.; Uhlmann, J.K. New extension of the Kalman filter to nonlinear systems. *Signal Process. Sens. Fusion Target Recognit. VI. Spie* **1997**, *3068*, 182–193.
51. Liu, Z.; Cai, Y.; Wang, H.; Chen, L.; Gao, H.; Jia, Y.; Li, Y. Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 6640–6653. [CrossRef]
52. Yin, C.; Jiang, H.; Tang, B.; Zhu, C.; Lin, Z.; Yin, Y. Handling Stability and Energy-Saving of Commercial Vehicle Electronically Controlled Hybrid Power Steering System. *J. Jiangsu Univ. Nat. Sci.* **2019**, *40*, 269.
53. Wang, H.; Ming, L.; Yin, C.; Long, C. Vehicle target detection algorithm based on fusion of lidar and millimeter wave radar. *J. Jiangsu Univ. Nat. Sci.* **2021**, *4*, 003.
54. Wang, H.; Chen, Z.; Cai, Y.; Chen, L.; Li, Y.; Sotelo, M.A.; Li, Z. Voxel-rcnn-complex: An effective 3-d point cloud object detector for complex traffic conditions. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–12. [CrossRef]
55. Wang, H.; Chen, Y.; Cai, Y.; Chen, L.; Li, Y.; Sotelo, M.A.; Li, Z. SFNet-N: An improved SFNet algorithm for semantic segmentation of low-light autonomous driving road scenes. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 21405–21417. [CrossRef]
56. Wang, H.; Cheng, L.; Yin, C.; Long, C.; You, H. Real-time visual vehicle detection method based on DSP platform. *J. Jiangsu Univ. Nat. Sci.* **2019**, *1*, 001.
57. Hermansdorfer, L.; Trauth, R.; Betz, J.; Lienkamp, M. End-to-end neural network for vehicle dynamics modeling. In Proceedings of the 2020 6th IEEE Congress on Information Science and Technology (CiSt), Agadir-Essaouira, Morocco, 5–12 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 407–412.
58. Xu, J.; Luo, Q.; Xu, K.; Xiao, X.; Yu, S.; Hu, J.; Miao, J.; Wang, J. An automated learning-based procedure for large-scale vehicle dynamics modeling on baidu apollo platform. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 5049–5056.
59. Nie, X.; Min, C.; Pan, Y.; Li, Z.; Królczyk, G. An Improved Deep Neural Network Model of Intelligent Vehicle Dynamics via Linear Decreasing Weight Particle Swarm and Invasive Weed Optimization Algorithms. *Sensors* **2022**, *22*, 4676. [CrossRef]