Satellite Navigation
https://satellite-navigation.springeropen.com/

ORIGINAL ARTICLE

Open Access

# Avnet: learning attitude and velocity for vehicular dead reckoning using smartphone by adapting an invariant EKF

Long Qian[1], Xinchuang Lin[1], Xiaoguang Niu[1], Qihai Huang[2], Leilei Li[2], Guangyi Guo[1], Zexin Wang[1] and Ruizhi Chen[1*]

## Abstract

Smartphone-based vehicle navigation has become the primary choice in everyday life due to low cost and real-time traffic updates. However, for vehicle navigation applications in the Global Navigation Satellite System (GNSS)-denied scenario such as parking lot and tunnel, it is quite difficult to maintain robust and continuous positioning based on consumer-grade sensors. In this paper, a novel method is proposed for accurate vehicular dead-reckoning based only on a smartphone inertial measurement unit. Robust vehicle dead reckoning can improve positioning performance in GNSS-degraded areas or where high-precision positioning sources are available at low-frequencies. The key components of the method are a Kalman filter with data-driven parameters adapter and a deep neural network that provides data-driven measurement estimation. A combined convolutional neural network and gated recurrent unit deep learning network, termed AVNet, is proposed to estimate the attitude and velocity of the vehicle. The learned measurements are integrated into an invariant Kalman filter to estimate Three-Dimensional (3D) attitude, velocity and position. The method was tested on custom datasets collected in a parking lot, and a 0.4 % relative horizontal translation error was achieved on average.

**Keywords** Inertial navigation, Smartphone, Inertial measurement unit, Deep learning, Invariant extended Kalman filter

## Introduction

Intelligent vehicles play an important role in the"Human × Car × Home"smart ecosystem. Although modern vehicles are equipped with in-vehicle satellite navigation systems, smartphones are still the more commonly used navigation terminal in daily life (Xu et al., 2021). Self-driving vehicles are usually equipped with multiple sensors such as Global Navigation Satellite System (GNSS) sensors, Inertial Measurement Units (IMUs), cameras, lidars, radars, and others (Wang et al., 2020). In comparison, many sensors are also built into smartphones, including GNSS (Zhang et al., 2024), Wi-Fi (Liu et al., 2022), Bluetooth Low Energy (BLE) (Li et al., 2022), cameras (Niu et al., 2023), gyroscopes, accelerometers (Guo et al., 2023), magnetometers (Niu et al., 2024), barometers (Wang et al., 2023a), microphones (Li et al., 2023), etc (Callebaut et al., 2019).

There are various types of sensors available for enabling location-based services, which are essential for the safety of self-driving vehicles. However, in GNSS-blocked areas, such as tunnels, underground parking garages, or multilevel flyovers (Gao et al., 2021), as well as under adverse weather conditions, the quality and accuracy of these services cannot be guaranteed. In

*Correspondence:
Ruizhi Chen
chenruizhi@cuhk.edu.cn
[1] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, No.129 Luoyu Road, Wuhan 430079, Hubei, China
[2] College of Aerospace Engineering, Chongqing University, No.174 Shazhengjie, Chongqing 400044, China

contrast, inertial and radar sensors are the only navigation technologies that can function in any location and under any weather and operational conditions. Among these, the IMU is a critical component of intelligent vehicles due to these advantages. Smartphone-grade IMUs offer several benefits, including low cost, lightweight design, and low power consumption. However, their accuracy and stability are considerably lower than those of automotive-grade IMUs, primarily due to limitations imposed by cost and form factor. Consequently, improving the accuracy and robustness of dead-reckoning using low-cost IMU has become a key research focus in the fields of mobile robotics and autonomous driving.

Artificial intelligence is increasingly powering change in every industry. Since IONet introduced the first Deep Neural Network (DNN) framework that reconstructs trajectories from raw inertial data and estimates uncertainties (Chen et al., 2021), there has been a growing interest in using Deep Learning (DL) to address the problem of inertial positioning. DL methods have demonstrated significant potential in addressing the challenges of pedestrian inertial navigation (Herath et al., 2020), but also can be applied to vehicle inertial navigation. AI-IMU dead-reckoning has demonstrated competitive performance which competes with the top-ranked lidars and stereo camera methods on the KITTI dataset (Brossard et al., 2020a). Integrating learned measurements and uncertainties can help reduce the drifts of inertial navigation mechanisms. Although the AI-IMU method works for high-precision IMUs, this method is not effective when applied to low-precision IMUs.

To constrain the unavoidable inertial integration error of Vehicle Dead Reckoning (VDR), in this paper a general framework is presented, the combined Data- and Model- Driven Vehicle Dead Reckoning (DMDVDR), which reconstructs accurate and robust trajectories from raw inertial measurements. The data-driven estimator Attitude Velocity Network (AVNet) is used to estimate the attitude and velocity pseudo-measurements, and a data-driven adapter is used to estimate the filter parameters. These two models constitute the data-driven part of the framework. The model-driven part consists of IMU modeling and vehicle dynamics. An Invariant Extended Kalman Filter (InEKF) is used to combine their output and estimate the attitude, velocity, and position of the vehicle. The main contributions of this paper are summarized as follows:

1) The DMDVDR method is presented, wherein the state-of-the-art InEKF is implemented to create an inertial positioning model that can leverage learned measurements to correct IMU integration. It yields accurate estimations of extended poses of the vehicle and the biases of the IMU, along with the associated uncertainty(covariance matrix).

2) We extend the approach in which inertial odometry has been combined with deep learning techniques. It is proposed to leverage learned attitude to correct system states. The data-driven module greatly improves the InEKF's robustness and accuracy.

3) We demonstrate the performance of the approach with experiments conducted in a realistic scenario. Experimental results show that high-precision VDR can be achieved with the integration of learned attitude and velocity measurements.

The remainder of this paper is organized as follows. In Sect. "Proposed method", some related works in vehicle positioning using smartphones and DL for inertial positioning are reviewed. In Sect. "Overview of proposed method" the proposed method is presented in detail. The data-driven and model-driven modules are presented in turn. In Sect."Data-driven part", the experimental setup is described, followed by the results and the associated analysis. The paper is concluded in Sect. "Data-driven filter parameter adapter".

## Related work

Research on smartphone-based vehicle navigation primarily focuses on multi-sensor fusion, with the main objective of enhancing the precision of GNSS positioning and overall navigation accuracy. However, there are inevitably scenarios where GNSS or other positioning signals are unavailable or unreliable. In such cases, the Inertial Navigation System (INS) plays a crucial role in maintaining navigation continuity. With the rapid advancement of DL methods and the increasing computational capabilities of smartphones, data-driven methods can now be implemented on these devices with real-time performance. Since the proposed framework consists of two components, the IMU-only dead-reckoning methods discussed in this work are categorized accordingly.

Model-driven methods typically incorporate sensors and vehicle dynamics models (Brossard et al., 2022; Zhang et al., 2020). Kalman filters represent a foundational approach in localization, which have been advanced in recent years. Among these, the error-state Kalman filter represents a major milestone in addressing dead-reckoning problems in which the error dynamic is slow (Solà, 2017). Furthermore, invariant Kalman filters are an improved variant over the plain Kalman filter and commonly employed in the field of localization, navigation, and simultaneous localization and mapping. Recent research shows extraordinary results in this domain (Barrau & Bonnabel, 2023), and the InEKF employed in the present framework is based on these advancements.

Data-driven vehicle dead reckoning methods significantly improve the robustness and accuracy of inertial navigation systems (Chen & Pan, 2024). Depending on whether a vehicle dynamic model is contained in the dead-reckoning pipeline, data-driven methods can be classified into two categories.

The first category pertains to the methods that learn the location displacement or velocity to obtain the positioning trajectories directly. Long Short-Term Memory (LSTM)-based DeepVIP-L and MobileNetV3-based XDRNet learned the velocity and heading from the gyroscope, accelerometer, magnetic field, and gravity sensors (Zhou et al., 2022a, b). Although the lightweight network used reduced the computational complexity, it did not achieve the same accuracy as data-driven pedestrian dead-reckoning methods. End-to-end methods achieve average errors of less than 10 ms in underground parking areas, but the velocity difference between vehicles and pedestrians may restrict these methods' generality. The other category includes the methods that learn pseudo-measurements and their uncertainties, which are then integrated with the vehicle's dynamic model. Based on the measurement type provided by inertial sensors, data-driven pseudo-measurements can be divided into pseudo-attitude and pseudo-velocity. The Bidirectional Long Short-Term Memory (BiLSTM)-based OriNet estimated Three-Dimensional (3D) attitude from the gyroscope measurements, the sampling time, and the prior state, and experiments on the EuRoC MAV dataset illustrated the power of this end-to-end learning approach (Esfahani et al., 2020). Brossard et al. (2020b) used a dilated Convolutional Neural Network(CNN) to denoise gyroscope sensor measurements and obtain attitude estimates through noise-free measurement integration. A BiLSTM neural network was used to predict the compensation parameters of the gyroscopes (Zhao et al., 2020). Apart from data-driven attitude measurements, data-driven velocity measurements can also be used to correct integration. A data-driven motion profile detector was proposed to identify specific parameters (Brossard et al., 2019), which can be incorporated in IMU integration as constraints, such as e.g. Zero Velocity Update (ZUPT), Zero Integrated Heading Rate (ZIHR), Non-Holonomic Constraint (NHC), and Odometry (ODO). The traveled distance or speed from odometry sensors in vehicles can be employed as a direct measurement to constrain vehicle motion, but they are not easily accessible on smartphones, so deep leaning-based pseudo-odometry has been proposed for the replacement of physical sensors, with notable examples such as OdoNet (Tang et al., 2022), SdoNet (Wang et al., 2023b), and DeepOdo (Wang et al., 2023a). DNNs were also used to adapt the noise parameters of the filter dynamically (Brossard et al., 2020a; Zhou et al., 2022c; Wang et al., 2023b). The experiments in Hong Kong with a HUAWEI Mate20 Pro have proven the feasibility of smartphone-based data-driven methods. However, the method does not perform as well in low-speed car park environments.

## Proposed method

### Overview of proposed method

In this section, the proposed method for vehicle navigation using smartphone IMU measurements only is described. The method belongs to deep learning-based inertial approaches which incorporate data-driven pseudo-measurements into the updating process of a Kalman filter for correcting IMU integration. The INS mechanization of the model-driven part is involved in the filter's propagation process, while the filter's update process incorporates data-driven measurements. There is no dedicated motion state estimation module for determining stationary states. The data-driven orientation and forward velocity with NHC act as continuous constraints that works in both stationary and moving state. The data-driven filter parameters are adapted dynamically at the same time. Fig. 1 illustrates the approach, which consists of three main blocks summarized as follows:

1. The InEKF integrates the inertial measurements (9)-(10) with the dynamic model (31) given by (34)-(42) in the propagating process, and exploits (56)-(58) as measurements (31) with the covariance matrix (54)-(55) to refine its estimates.
2. AVNet (1) estimates the attitude and velocity by learning from a single IMU. The DNN of these two types of measurements' outputs shares the same architecture. For attitude measurements, the output of the proposed AVNet is the change in attitude so that parameter singularities, gimbal lock issues, or the like are avoided. The changes in attitude are then integrated to get attitude measurements.
3. The filter parameter adapter determines the corresponding covariance matrix. This DL-based adapter learns filter parameters from only raw IMU measurements without the information about any filter state or any other quantity.

### Data-driven part

#### *Data-driven measurement estimator AVNet*

The motivation for data-driven measurements comes from the sensors that self-driving vehicles are already equipped with. The forward velocity measurements obtained from wheeled odometry sensors can be composed with the NHC to perform 3D auxiliary velocity updates to realize dynamic constraints on the vehicle's

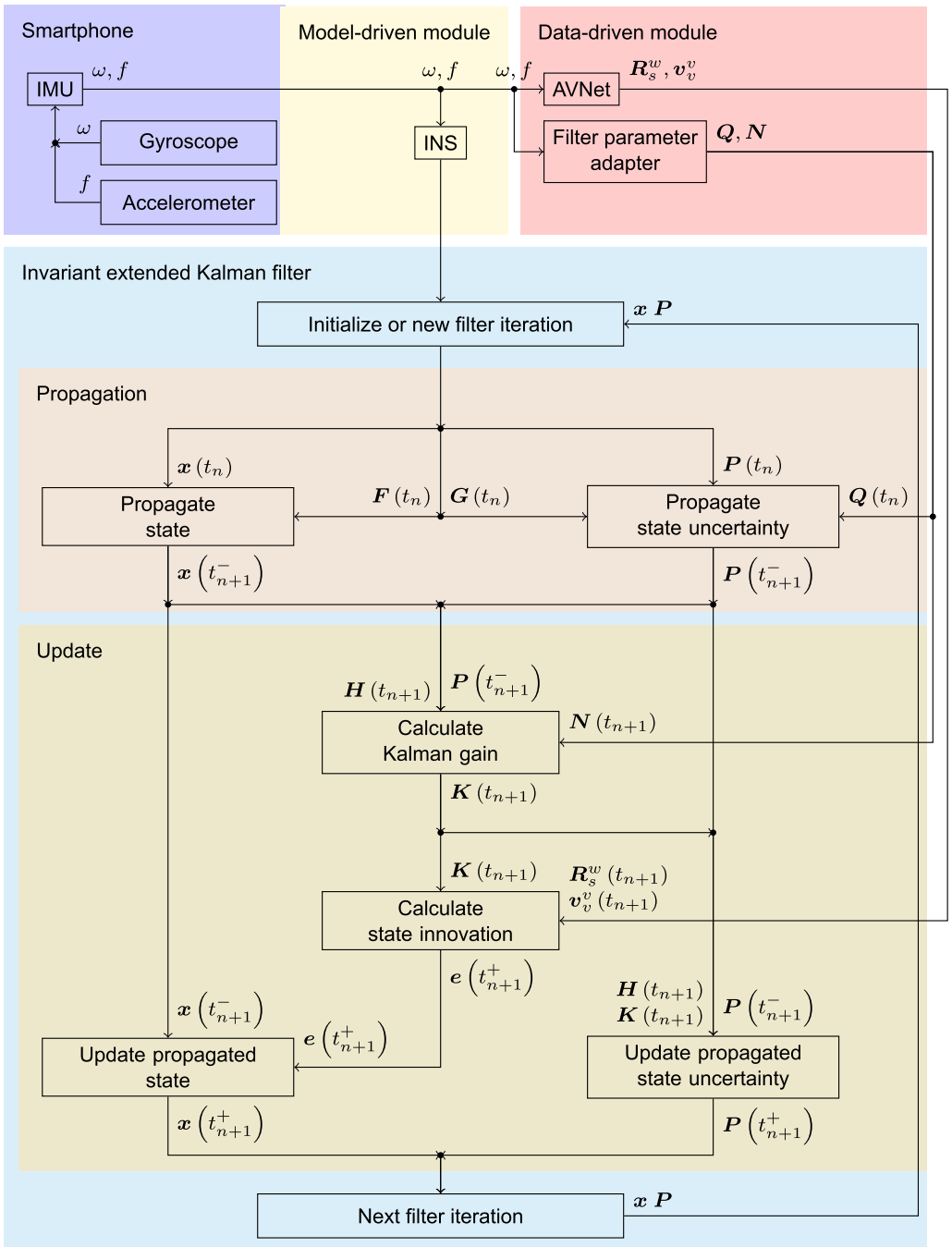


**Fig. 1** The framework of the proposed DMDVDR method

velocity (Niu et al., 2007). In addition, steering wheel angle sensors can provide rotation measurements for correcting the lateral motion model. Those auxiliary measurements from the on-board sensors are available to the in-vehicle navigation algorithms via the controller area network. While it is usually limited for smartphones to access in-vehicle sensors' data. Hence, the deep learning virtual sensors become a feasible approach to realize functions similar to physical sensors. The measurement estimator calculates the attitude and forward velocity at each instant $t_n$. The inputs to the estimator are IMU observations consisting of angular velocity $\tilde{\boldsymbol{\omega}}$ and specific forces $\tilde{\boldsymbol{f}}$, while the output is:

$$\boldsymbol{R}_s^w(t_n), v_{v,\text{lon}}^v(t_n) = f_{\text{Estimator}}\left(\{\tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{f}}\}_{n-W_{\text{est}}+1}^{n}\right) \quad (1)$$

where $\boldsymbol{R}_s^w$ is the Data-Driven Attitude (DDATT) and $v_{v,\text{lon}}^v$ is the Data-Driven Velocity (DDODO) which is call DDODO in this paper, while $W_{\text{est}}$ is the size of input window. $W_{\text{est}}$ is set to 200, so that the output frequency of AVNet is 1 Hz in the experiments.

Recent studies have shown that hybrid network architectures can synthesize the advantages of single-structure networks to achieve better performance (Wang et al., 2022). The main structure of the proposed AVNet is a combined CNN-Gated Recurrent Unit(GRU) module which is shown in Fig. 2. Hidden layer names are labeled in the upper part of the diagram. Kernel sizes of convolution layers and max pooling layers are labeled above the arrows. Sizes of input, output and features are labeled along the boxes. Colors of boxes correspond to layer names. Hybrid network architectures have higher accuracy than single ones (Wang et al., 2023a). Also, we have solved the sampling problems in data preprocessing. The input data size of a single window for AVNet is (200, 6). For DDATT, the output size of AVNet is (3), while the output size is (1) for DDODO. AVNet consists of two 1D-convolution layers with Rectified Linear Unit (ReLU) and max-pooling layers. The features extracted by the CNN are passed through two fully-connected layers to the GRU modules. Finally, the learned measurements are regressed from the GRU modules.

For DDATT's attitude calculation, the loss function of the proposed network is defined as the mean square error of the predicted attitude change and the reference ground truth change representing by the quaternion's $x$, $y$, $z$ values.

$$L_{\boldsymbol{R}_s^w} = \frac{1}{N}\sum_{i=1}^{N}\left(\Delta\boldsymbol{q}_i^{\text{pred}} - \Delta\boldsymbol{q}_i^{\text{true}}\right)^2 \quad (2)$$

where $N$ is the number of clipped time sections, $\Delta\boldsymbol{q}_i^{\text{pred}}$ and $\Delta\boldsymbol{q}_i^{\text{true}}$ are the estimated and ground truth attitude change of the sensor represented by the quaternions. Since the rotation is represented by the unit quaternions, the complete expression for the unit quaternions can be computed from the $x$, $y$, $z$ components. And the final attitude measurement is obtained by accumulating

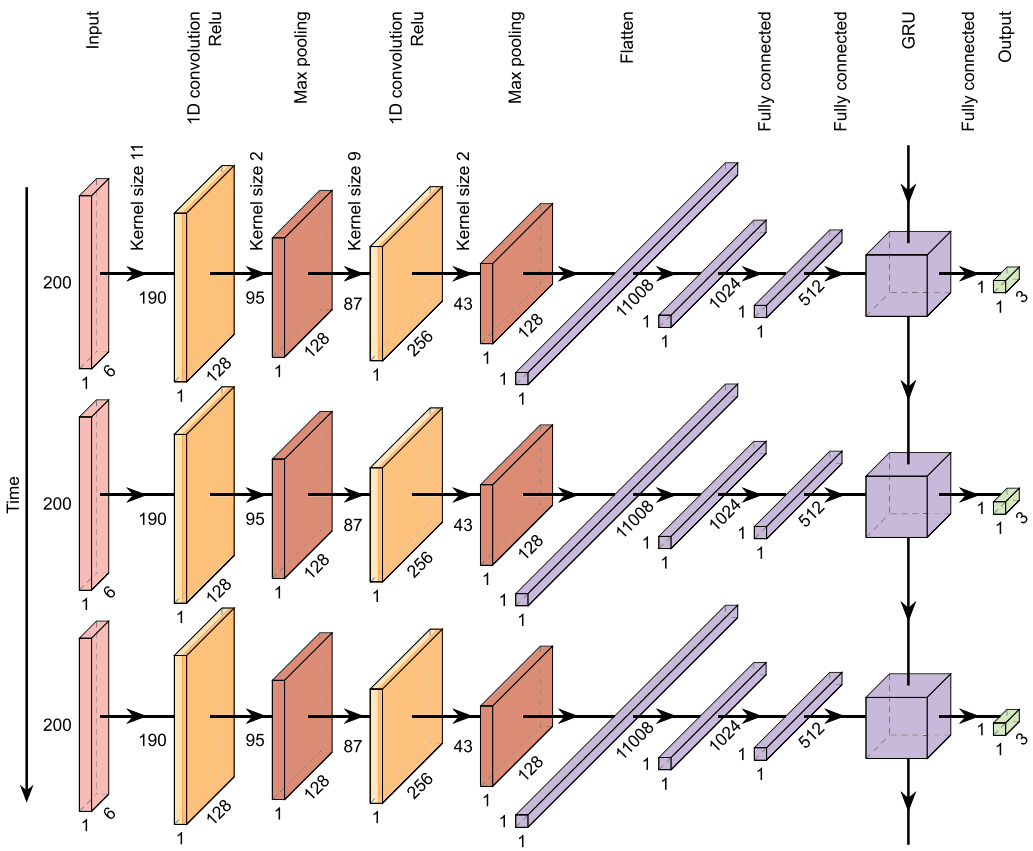


**Fig. 2** Architecture of the proposed AVNet

the predicted attitude change. Initial attitude need to be given for integrating the attitude measurements.

For DDODO, the loss function of AVNet is defined as the mean square error of the predicted velocity and the reference ground truth velocity (Wang et al., 2023a).

$$L_{v_v^v} = \frac{1}{N}\sum_{i=1}^{N}\left(v_i^{\text{pred}} - v_i^{\text{true}}\right)^2 \tag{3}$$

where $N$ is the number of velocity training samples, while $v_i^{\text{pred}}$ and $v_i^{\text{true}}$ are the estimated and ground truth velocities of the vehicle.

#### *Data-driven filter parameter adapter*

The measurement filter parameter adapter calculates the process and measurement covariances $\boldsymbol{Q}(t_n)$ and $\boldsymbol{N}(t_{n+1})$, respectively, at each instant $t_n$. The core architecture of the adapter network is that of a CNN. The adapter's input is also a window of $W_{\text{adapter}}$ inertial sensor measurements and its output is

$$\boldsymbol{Q}(t_n), \boldsymbol{N}(t_{n+1}) = f_{\text{adapter}}\left(\{\tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{f}}\}_{n-W_{\text{adapter}}+1}^{n}\right) \tag{4}$$

In terms of implementation, the simple CNN-like architecture shown in Fig. 3 is adopted to let the network be trainable. Also a simple network structure avoids overfitting, while similar network structures have been validated by Brossard et al. (2020a). The labels and boxes have the same meaning as Fig. 2. $W_{\text{adapter}}$ is equivalent to 20 in this architecture. The output frequency of the Adapter is 200 Hz.

The parameters of the measurement filter were trained using an indirect optimization process. Specifically, the outputs of the adapter were integrated into the InEKF, and the adapter was optimized by minimizing a loss function based on relative translation errors. The parameters obtained from the adapter are closer to their mathematical meaning without physical one.

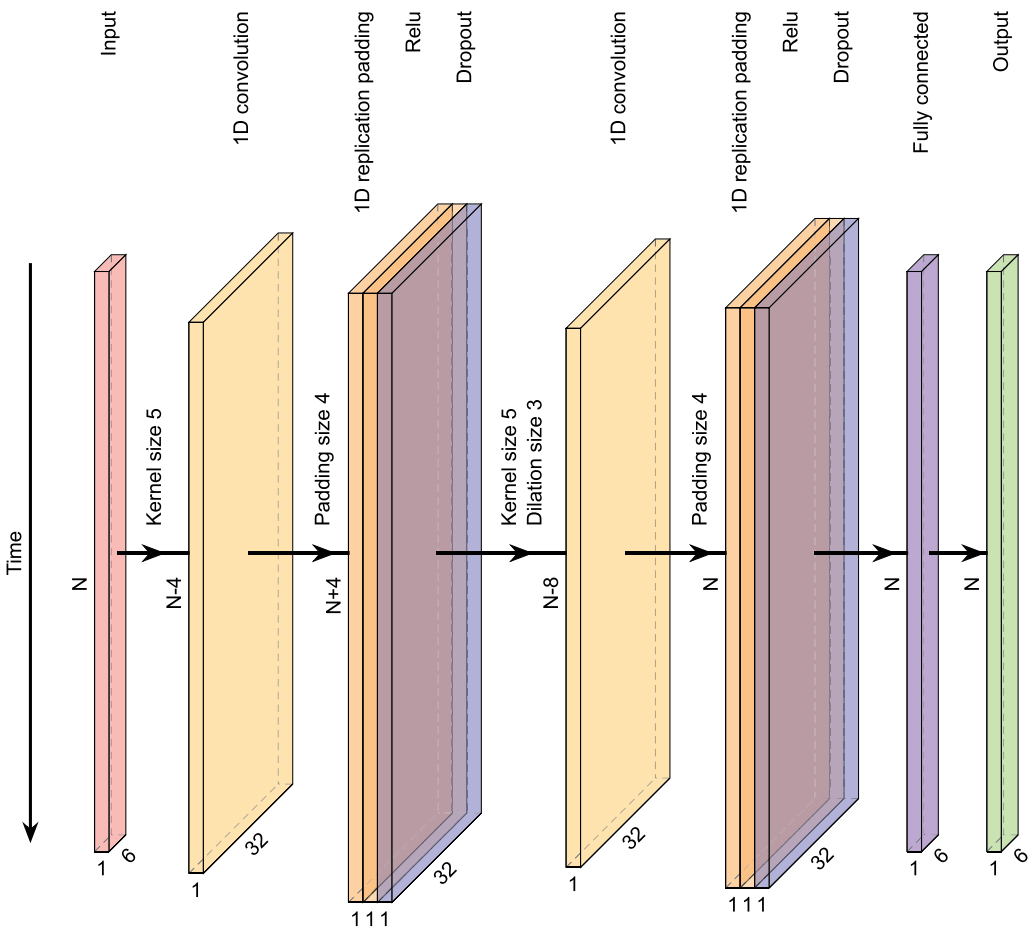


**Fig. 3** Architecture of the proposed filter parameter adapter

#### *Implementation details*

The full approach of the data-driven part was implemented in Python using the PyTorch library. AVNet's implementation has been published as open source and is available at: https://github.com/DragonEmperorG/QDeepOdo. The network's implementation is similar to the DeepOdo network (Wang et al., 2023a), but the sampling rate is different and the barometer data are not considered.

The implementation of the data-driven filter parameter adapter network has also been published at: https://github.com/DragonEmperorG/QAIIMUDeadReckoning. The adapter's implementation is similar to that of the AI-IMU method except the output. The adapter network's output is modified based on the data-driven measuremet parameters' number. The adapter is a 2-layer CNN, with each layer consisting of a 1D CNN, a replication layer, a ReLU and a dropout layer. The first layer has kernel size 5, input dimention 6, output dimention 32, and dilation parameter 1. The second layer has kernel size 5, input dimention 32, output dimention 32, and dilation parameter 3. The filter parameter scalar vectors $(q_{\mathrm{s,a}}, n_{\mathrm{s,a}})$ are regressed by the linear layer. Then, the process noise parameter $Q$ and the measurement noise parameter $N$ are calculated as:

$$\begin{aligned}
\boldsymbol{Q} &= f_{\mathrm{diag}}\left(\boldsymbol{Q}_{\boldsymbol{\omega}_s^s}, \boldsymbol{Q}_{\boldsymbol{f}_s^s}, \boldsymbol{Q}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}}, \boldsymbol{Q}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}}, \boldsymbol{Q}_{\boldsymbol{R}_s^v}, \boldsymbol{Q}_{\boldsymbol{p}_v^s}\right) \\
\boldsymbol{Q}_{\boldsymbol{\omega}_s^s} &= \left(Q_{\boldsymbol{\omega}_s^s,x}\ Q_{\boldsymbol{\omega}_s^s,y}\ Q_{\boldsymbol{\omega}_s^s,z}\right) \\
\boldsymbol{Q}_{\boldsymbol{f}_s^s} &= \left(Q_{\boldsymbol{f}_s^s,x}\ Q_{\boldsymbol{f}_s^s,y}\ Q_{\boldsymbol{f}_s^s,z}\right) \\
\boldsymbol{Q}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}} &= \left(Q_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s},x}\ Q_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s},y}\ Q_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s},z}\right) \\
\boldsymbol{Q}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}} &= \left(Q_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s},x}\ Q_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s},y}\ Q_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s},z}\right) \\
\boldsymbol{Q}_{\boldsymbol{R}_s^v} &= \left(Q_{\boldsymbol{R}_s^v,x}\ Q_{\boldsymbol{R}_s^v,y}\ Q_{\boldsymbol{R}_s^v,z}\right) \\
\boldsymbol{Q}_{\boldsymbol{p}_v^s} &= \left(Q_{\boldsymbol{p}_s^v,x}\ Q_{\boldsymbol{p}_s^v,y}\ Q_{\boldsymbol{p}_s^v,z}\right)
\end{aligned} \tag{5}$$

$$\begin{aligned}
Q_{\mathrm{s,a}} &= \left(\sigma_{\mathrm{s,a}}\right)^2\left(10^{\beta \tanh\left(q_{\mathrm{s,a}}\right)}\right) \\
&\mathrm{s} \in \left\{\boldsymbol{\omega}_s^s, \boldsymbol{f}_s^s, \boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}, \boldsymbol{\delta}_{\boldsymbol{f}_s^s}, \boldsymbol{R}_s^v, \boldsymbol{p}_v^s\right\}, \mathrm{a} \in \{x, y, z\}
\end{aligned} \tag{6}$$

where the matrix $Q$ is a diagonal matrix whose diagonal elements correspond to the states in (32)-(33). There are 6 process noise states, and the default is: $\sigma_{\boldsymbol{\omega}_s^s,\mathrm{a}} = 1 \times 10^{-2}\,\mathrm{rad/s}$, $\sigma_{\boldsymbol{f}_s^s,\mathrm{a}} = 1 \times 10^{-2}\,\mathrm{rad/s^2}$, $\sigma_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s},\mathrm{a}} = 1 \times 10^{-4}\,\mathrm{rad/s}$, $\sigma_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s},\mathrm{a}} = 1 \times 10^{-3}\,\mathrm{rad/s^2}$, $\sigma_{\boldsymbol{R}_s^v,\mathrm{a}} = 1 \times 10^{-4}\mathrm{rad}$, $\sigma_{\boldsymbol{p}_v^s,\mathrm{a}} = 1 \times 10^{-4}\,\mathrm{m}$, $\mathrm{a} \in \{x, y, z\}$.

$$\begin{aligned}
\boldsymbol{N} &= f_{\mathrm{diag}}\left(\boldsymbol{N}_{\boldsymbol{R}_s^w}, \boldsymbol{N}_{\boldsymbol{v}_s^v}\right) \\
\boldsymbol{N}_{\boldsymbol{R}_s^w} &= \left(N_{\boldsymbol{R}_s^w,x}\ N_{\boldsymbol{R}_s^w,y}\ N_{\boldsymbol{R}_s^w,z}\right) \\
\boldsymbol{N}_{\boldsymbol{v}_v^s} &= \left(N_{\boldsymbol{v}_s^v,x}\ N_{\boldsymbol{v}_s^v,y}\ N_{\boldsymbol{v}_s^v,z}\right)
\end{aligned} \tag{7}$$

$$\begin{aligned}
N_{\mathrm{s,a}} &= \left(\sigma_{\mathrm{s,a}}\right)^2\left(10^{\beta \tanh\left(n_{\mathrm{s,a}}\right)}\right) \\
&\mathrm{s} \in \left\{\boldsymbol{R}_s^w, \boldsymbol{v}_s^v\right\}, \mathrm{a} \in \{x, y, z\}
\end{aligned} \tag{8}$$

where the matrix $N$ is also a diagonal matrix, whose elements correspond to the states in (54)-(55). There are 6 process noise states, and the default is: $\sigma_{\boldsymbol{R}_s^w,\mathrm{a}} = 1 \times 10^{-2}\,\mathrm{rad/s}$, $\sigma_{\boldsymbol{v}_s^v,\mathrm{a}} = 1\,\mathrm{m/s}$, $\mathrm{a} \in \{x, y, z\}$. We define parameter $\beta = 3$ allowing for each covariance element to be $1 \times 10^3$ bigger or $1 \times 10^{-3}$ smaller than its default value.

#### *Training*

Training was performed on a server with an RTX 4090 GPU, and the different data-driven parts were trained independently.

First, AVNet was trained using the Adam optimizer with learning rate of 0.0001 and a batch size of 1024. For the adapter network, the same routines in the AI-IMU method were used for training, with the optimization objective being the relative translation error calculated from the filter estimates.

### Model-driven part

#### *Vehicle dead reckoning dynamics*

The only input measurements available were those from the smartphone IMU sensor, which is the same as AVNet:

$$\tilde{\boldsymbol{\omega}}_s^s(t) = \boldsymbol{\omega}_s^s(t) + \boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}(t) + \boldsymbol{w}_{\boldsymbol{\omega}_s^s}(t) \tag{9}$$

$$\tilde{\boldsymbol{f}}_s^s(t) = \boldsymbol{f}_s^s(t) + \boldsymbol{\delta}_{\boldsymbol{f}_s^s}(t) + \boldsymbol{w}_{\boldsymbol{f}_s^s}(t) \tag{10}$$

where $\boldsymbol{\omega}_s^s(t)$, $\boldsymbol{f}_s^s(t)$ are true angular velocity and specific forces. $\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}(t)$, $\boldsymbol{\delta}_{\boldsymbol{f}_s^s}(t)$ are true quasi-constant biases. $\boldsymbol{w}_{\boldsymbol{\omega}_s^s}(t)$ and $\boldsymbol{w}_{\boldsymbol{f}_s^s}(t)$ are zero-mean Gaussian noise; and $\cdot(t)$ represents the value of a measurement at the instantaneous time $t$.

The individual terms of the system dynamics in continuous time are expressed as:

$$\dot{\boldsymbol{R}}_s^w(t) = \boldsymbol{R}_s^w(t)\left(\boldsymbol{\omega}_s^s(t)\right)_\times \tag{11}$$

$$\dot{\boldsymbol{v}}_s^w(t) = \boldsymbol{R}_s^w(t)\boldsymbol{f}_s^s(t) + \boldsymbol{g}_s^w \tag{12}$$

$$\dot{\boldsymbol{p}}_s^w(t) = \boldsymbol{v}_s^w(t) \tag{13}$$

$$\dot{\boldsymbol{\delta}}_{\omega_s^s}(t) = \boldsymbol{w}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}}(t) \tag{14}$$

$$\dot{\boldsymbol{\delta}}_{f_s^s}(t) = \boldsymbol{w}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}}(t) \tag{15}$$

$$\dot{\boldsymbol{R}}_s^v(t) = \boldsymbol{R}_s^v(t)\left(\boldsymbol{w}_{\boldsymbol{R}_s^v}(t)\right)_\times \tag{16}$$

$$\dot{\boldsymbol{p}}_v^s(t) = \boldsymbol{w}_{\boldsymbol{p}_v^s}(t) \tag{17}$$

There are three coordinate systems used in the paper, namely the world, sensor and vehicle coordinate systems. The variables follow the pattern $\cdot_{subscript}^{superscript}$, where the superscript denotes the reference coordinate system while the subscript denotes the object coordinate system. In the world coordinate system, the attitude, velocity and position of a sensor are denoted using $\boldsymbol{R}_s^w \in \mathbf{G}_{\mathrm{SO(3)}}$, $\boldsymbol{v}_s^w \in \mathbb{R}^3$ and $\boldsymbol{p}_s^w \in \mathbb{R}^3$, respectively. To be specific, $\boldsymbol{R}_s^w$ maps the sensor coordinate to the world coordinate. $(\cdot)_\times : \mathbb{R}^3 \to \mathbf{g}_{\mathrm{so(3)}}$ denotes the skew symmetric matrix, which maps a vector to the corresponding element of the Lie algebra $\mathbf{g}_{\mathrm{so(3)}}$. In the sensor coordinate system, the true angular velocity and acceleration are defined as $\boldsymbol{\omega}_s^s \in \mathbb{R}^3$ and $\boldsymbol{f}_s^s \in \mathbb{R}^3$, respectively. They should be substituted using equations (18) and (19). The IMU bias dynamics are modeled as the zero-mean white Gaussian processes $\boldsymbol{w}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}}(t)$ and $\boldsymbol{w}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}}(t)$. In the vehicle coordinate system, to express the DL based measurement, the attitude of the sensor is set to $\boldsymbol{R}_s^v \in \mathbf{G}_{\mathrm{SO(3)}}$, which maps the sensor coordinates to the vehicle coordinates. At the same time, $\boldsymbol{p}_v^s \in \mathbb{R}^3$ is introduced to represent the lever arm vector from the vehicle coordinate origin to the sensor coordinate origin in the sensor coordinate system. In the experiment, the smartphone was rigidly attached to the vehicle, so $\boldsymbol{R}_s^v$ and $\boldsymbol{p}_v^s$ were approximately constant. $\boldsymbol{w}_{\boldsymbol{R}_s^v}(t)$ and $\boldsymbol{w}_{\boldsymbol{p}_v^s}(t)$ are the random impulses applied to the attitude and position, which are modeled using a white Gaussian process. Their mean is zero, and their covariance will be determined from the filter parameter adapter.

The target scenario in this paper involved an underground parking lot whose size is limited, so the earth is considered flat, and gravity $\boldsymbol{g}_s^w \in \mathbb{R}^3$ in the world coordinate system is assumed to be a known constant. Owing to the precision of smartphone IMU sensors, the effects of earth rotation and Coriolis acceleration are ignored.

$$\boldsymbol{\omega}_s^s = \tilde{\boldsymbol{\omega}}_s^s(t) - \boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}(t) - \boldsymbol{w}_{\boldsymbol{\omega}_s^s}(t) \tag{18}$$

$$\boldsymbol{f}_s^s = \tilde{\boldsymbol{f}}_s^s(t) - \boldsymbol{\delta}_{\boldsymbol{f}_s^s}(t) - \boldsymbol{w}_{\boldsymbol{f}_s^s}(t) \tag{19}$$

#### *The invariant extended Kalman filter*

The InEKF's convergence and stability make it particularly suitable for navigation applications (Wang et al., 2020; Bai et al., 2024). It has currently been employed for the implementation of several industrial applications (Barrau & Bonnabel, 2017). However, the InEKF has not yet been widely applied in the field of smartphone in-vehicle navigation. In this paper, an InEKF is adapted to perform the fusion of the inertial measurements and pseudo-measurements. Additionally, the new pseudo-measurements of DDATT are added to the InEKF which incorporate with DDODO and Data-Driven Non-Holonomic Constraint (DDNHC) to realize Six-Dimensional (6D) geometric constraints. The filter state defines as a tuple of five variables.

$$\boldsymbol{x} = \left(\boldsymbol{\chi}_s^w, \boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}, \boldsymbol{\delta}_{\boldsymbol{f}_s^s}, \boldsymbol{R}_s^v, \boldsymbol{p}_v^s\right) \tag{20}$$

The variable $\boldsymbol{\chi}_s^w$ is embedded in the Lie group $\mathbf{G}_{\mathrm{SE}_2(3)}$, $\boldsymbol{R}_s^v$ is embedded in the Lie group $\mathbf{G}_{\mathrm{SO(3)}}$, $\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}$, $\boldsymbol{\delta}_{\boldsymbol{f}_s^s}$ and $\boldsymbol{p}_v^s$ are treated as vectors.

$$\boldsymbol{\chi}_s^w = \begin{pmatrix} \boldsymbol{R}_s^w & \boldsymbol{v}_s^w & \boldsymbol{p}_s^w \\ \mathbf{0}_{1\times 3} & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 \end{pmatrix} \tag{21}$$

Correspondingly $\boldsymbol{\eta}_{\boldsymbol{\chi}_s^w}$ and $\boldsymbol{\eta}_{\boldsymbol{R}_s^v}$ are defined as right-invariant errors, while $\boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}}$, $\boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}}$ and $\boldsymbol{\xi}_{\boldsymbol{p}_v^s}$ are defined as distinct linear errors. The state error $\boldsymbol{e}$ can be regarded as a composite error.

$$\boldsymbol{e} = \left(\boldsymbol{\eta}_{\boldsymbol{\chi}_s^w}, \boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}}, \boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}}, \boldsymbol{\eta}_{\boldsymbol{R}_s^v}, \boldsymbol{\xi}_{\boldsymbol{p}_v^s}\right) \tag{22}$$

Written explicitly, the right-invariant error $\boldsymbol{\eta}_{\boldsymbol{\chi}_s^w}$ can be expanded as follows:

$$\boldsymbol{\eta}_{\boldsymbol{\chi}_s^w} = \hat{\boldsymbol{\chi}}_s^w\left(\boldsymbol{\chi}_s^w\right)^{-1} = \exp_{\mathbf{G}_{\mathrm{SE}_2(3)}}\left(\boldsymbol{\xi}_s^w\right) = \begin{pmatrix} \boldsymbol{\eta}_{\boldsymbol{R}_s^w} & \boldsymbol{\xi}_{\boldsymbol{v}_s^w} & \boldsymbol{\xi}_{\boldsymbol{p}_s^w} \\ \mathbf{0}_{1\times 3} & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 \end{pmatrix} \tag{23}$$

Where $\boldsymbol{\xi}_s^w \in \mathbb{R}^9$ are mapped to the Lie group $\mathrm{SE}_2(3)$. This mapping follows two steps: a mapping to the Lie algebra $\mathbf{g}_{\mathrm{se}_2(3)}$ is first created and then the exponential of $\mathbf{G}_{\mathrm{SE}_2(3)}$ is employed. Furthermore, $\boldsymbol{\xi}_s^w \in \mathbb{R}^9$ comprises three parts, which can be written separately, while $\hat{(\cdot)}$ represents the state estimation.

$$\boldsymbol{\eta}_{\boldsymbol{R}_s^w} = \hat{\boldsymbol{R}}_s^w\left(\boldsymbol{R}_s^w\right)^{\mathrm{T}} = \exp_{\mathbf{G}_{\mathrm{SO(3)}}}\left(\boldsymbol{\xi}_{\boldsymbol{R}_s^w}\right) \tag{24}$$

$$\boldsymbol{\xi}_{\boldsymbol{v}_s^w} = \hat{\boldsymbol{v}}_s^w - \hat{\boldsymbol{R}}_s^w\left(\boldsymbol{R}_s^w\right)^{\mathrm{T}}\boldsymbol{v}_s^w \tag{25}$$

$$\boldsymbol{\xi}_{\boldsymbol{p}_s^w} = \hat{\boldsymbol{p}}_s^w - \hat{\boldsymbol{R}}_s^w\left(\boldsymbol{R}_s^w\right)^{\mathrm{T}}\boldsymbol{p}_s^w \tag{26}$$

$$\boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}} = \boldsymbol{\delta}_{\boldsymbol{\omega}_s^s} - \hat{\boldsymbol{\delta}}_{\boldsymbol{\omega}_s^s} \tag{27}$$

$$\boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}} = \boldsymbol{\delta}_{\boldsymbol{f}_s^s} - \hat{\boldsymbol{\delta}}_{\boldsymbol{f}_s^s} \tag{28}$$

$$\boldsymbol{\eta}_{\boldsymbol{R}_s^v} = \hat{\boldsymbol{R}}_s^v \left(\boldsymbol{R}_s^v\right)^{-1} = \exp_{\mathbf{G}_{\mathrm{SO}(3)}}\left(\boldsymbol{\xi}_{\boldsymbol{R}_s^v}\right) \tag{29}$$

$$\boldsymbol{\xi}_{\boldsymbol{p}_v^s} = \boldsymbol{p}_v^s - \hat{\boldsymbol{p}}_v^s \tag{30}$$

where $\boldsymbol{\xi}_{\boldsymbol{R}_s^w}$ and $\boldsymbol{\xi}_{\boldsymbol{R}_s^v}$ are mapped to the Lie group $\mathbf{G}_{\mathrm{SO}(3)}$.

In the extended Kalman filter, the abstract form of the state transition model can be written as:

$$\boldsymbol{x}(t_{n+1}) = f(\boldsymbol{x}(t_n), \boldsymbol{u}(t_n), \boldsymbol{w}(t_n)) \tag{31}$$

where $\cdot(t_n)$ denotes discrete time, $\boldsymbol{x}$ denotes the filter state (20), $\boldsymbol{u}$ is the control vector, and $\boldsymbol{w}$ is the process noise, which is predefined locally (9) (10) and (14)-(17). Here, the global form can be written as:

$$\boldsymbol{w} = \left(\left(\boldsymbol{w}_{\boldsymbol{\omega}_s^s}\right)^{\mathrm{T}}\left(\boldsymbol{w}_{\boldsymbol{f}_s^s}\right)^{\mathrm{T}}\left(\boldsymbol{w}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}}\right)^{\mathrm{T}}\left(\boldsymbol{w}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}}\right)^{\mathrm{T}}\left(\boldsymbol{w}_{\boldsymbol{R}_s^v}\right)^{\mathrm{T}}\left(\boldsymbol{w}_{\boldsymbol{p}_v^s}\right)^{\mathrm{T}}\right)^{\mathrm{T}} \tag{32}$$

$$\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}_{18\times 1}, \boldsymbol{Q}) \tag{33}$$

where $\boldsymbol{Q}$ is the process covariance.

From the continuous system dynamics (11)-(17), the individual terms of the discrete dynamics can be written as:

$$\boldsymbol{u}_{\omega_s^s}(t_n) = \tilde{\boldsymbol{\omega}}_s^s(t_n) - \hat{\boldsymbol{\delta}}_{\omega_s^s}(t_n) \tag{34}$$

$$\boldsymbol{u}_{f_s^s}(t_n) = \tilde{\boldsymbol{f}}_s^s(t_n) - \hat{\boldsymbol{\delta}}_{f_s^s}(t_n) \tag{35}$$

$$\hat{\boldsymbol{R}}_s^w\left(t_{n+1}^-\right) = \hat{\boldsymbol{R}}_s^w(t_n)\exp\left(\left(\boldsymbol{u}_{\omega_s^s}(t_n)\mathrm{d}t\right)_\times\right) \tag{36}$$

$$\hat{\boldsymbol{v}}_s^w\left(t_{n+1}^-\right) = \hat{\boldsymbol{v}}_s^w(t_n) + \left(\hat{\boldsymbol{R}}_s^w(t_n)\boldsymbol{u}_{f_s^s}(t_n) + \boldsymbol{g}\right)\mathrm{d}t \tag{37}$$

$$\hat{\boldsymbol{p}}_s^w\left(t_{n+1}^-\right) = \hat{\boldsymbol{p}}_s^w(t_n) + \hat{\boldsymbol{v}}_s^w(t_n)\mathrm{d}t \tag{38}$$

$$\hat{\boldsymbol{\delta}}_{\boldsymbol{\omega}_s^s}\left(t_{n+1}^-\right) = \hat{\boldsymbol{\delta}}_{\boldsymbol{\omega}_s^s}(t_n) \tag{39}$$

$$\hat{\boldsymbol{\delta}}_{\boldsymbol{f}_s^s}\left(t_{n+1}^-\right) = \hat{\boldsymbol{\delta}}_{\boldsymbol{f}_s^s}(t_n) \tag{40}$$

$$\hat{\boldsymbol{R}}_s^v\left(t_{n+1}^-\right) = \hat{\boldsymbol{R}}_s^v(t_n) \tag{41}$$

$$\hat{\boldsymbol{p}}_v^s\left(t_{n+1}^-\right) = \hat{\boldsymbol{p}}_v^s(t_n) \tag{42}$$

where $\cdot\left(t_{n+1}^-\right)$ denotes the propagated state on discrete time. The IMU's measurement between $t_n$ and $t_{n+1}$ is assumed to be a zero-order hold. The derivatives of the composite errors are given separately,

$$\dot{\boldsymbol{\eta}}_{\boldsymbol{R}_s^w} = \left(\dot{\boldsymbol{\xi}}_{\boldsymbol{R}_s^w}\right)_\times \approx \left(\hat{\boldsymbol{R}}_s^w\left(\boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}} + \boldsymbol{w}_{\boldsymbol{\omega}_s^s}\right)\right)_\times \tag{43}$$

$$\dot{\boldsymbol{\xi}}_{\boldsymbol{v}_s^w} \approx (\boldsymbol{g})_\times \boldsymbol{\xi}_{\boldsymbol{R}_s^w} + \left(\left(\boldsymbol{v}_s^w\right)_\times \hat{\boldsymbol{R}}_s^w\right)\left(\boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}} + \boldsymbol{w}_{\boldsymbol{\omega}_s^s}\right) + \hat{\boldsymbol{R}}_s^w\left(\boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}} + \boldsymbol{w}_{\boldsymbol{f}_s^s}\right) \tag{44}$$

$$\dot{\boldsymbol{\xi}}_{\boldsymbol{p}_s^w} \approx \boldsymbol{\xi}_{\boldsymbol{v}_s^w} + \left(\left(\boldsymbol{p}_s^w\right)_\times \hat{\boldsymbol{R}}_s^w\right)\left(\boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}} + \boldsymbol{w}_{\boldsymbol{\omega}_s^s}\right) \tag{45}$$

$$\dot{\boldsymbol{\xi}}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}} = \boldsymbol{w}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}} \tag{46}$$

$$\dot{\boldsymbol{\xi}}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}} = \boldsymbol{w}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}} \tag{47}$$

$$\dot{\boldsymbol{\eta}}_{\boldsymbol{R}_s^v} = \left(\dot{\boldsymbol{\xi}}_{\boldsymbol{R}_s^v}\right)_\times \approx \left(\hat{\boldsymbol{R}}_s^v \boldsymbol{w}_{\boldsymbol{R}_s^v}\right)_\times \tag{48}$$

$$\dot{\boldsymbol{\xi}}_{\boldsymbol{p}_v^s} = \boldsymbol{w}_{\boldsymbol{p}_v^s} \tag{49}$$

The covariance matrix is updated through

$$\boldsymbol{P}\left(t_{n+1}^-\right) = \boldsymbol{F}(t_n)\boldsymbol{P}(t_n)(\boldsymbol{F}(t_n))^{\mathrm{T}} + \boldsymbol{G}(t_n)\boldsymbol{Q}(t_n)(\boldsymbol{G}(t_n))^{\mathrm{T}} \tag{50}$$

where $\boldsymbol{F}$ and $\boldsymbol{G}$ are the Jacobian matrix of (31) with respect to $\boldsymbol{x}$ and $\boldsymbol{w}$. From (43)-(49), they can be calculated by solving the Riccati equation, and one approximate solution is given as:

$$\boldsymbol{F} = \boldsymbol{I}_{21} + \begin{pmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \hat{\boldsymbol{R}}_s^w & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ (\boldsymbol{g})_\times & \mathbf{0}_3 & \mathbf{0}_3 & \left(\boldsymbol{v}_s^w\right)_\times \hat{\boldsymbol{R}}_s^w & \hat{\boldsymbol{R}}_s^w & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \left(\boldsymbol{p}_s^w\right)_\times \hat{\boldsymbol{R}}_s^w & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ & & & \mathbf{0}_{12\times 21} & & & \end{pmatrix} \mathrm{d}t \tag{51}$$

$$\boldsymbol{G} = \begin{pmatrix} \hat{\boldsymbol{R}}_s^w & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \left(\boldsymbol{v}_s^w\right)_\times \hat{\boldsymbol{R}}_s^w & \hat{\boldsymbol{R}}_s^w & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \left(\boldsymbol{p}_s^w\right)_\times \hat{\boldsymbol{R}}_s^w & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \hat{\boldsymbol{R}}_s^v & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 \end{pmatrix} \mathrm{d}t \tag{52}$$

where $\boldsymbol{I}_n$ and $\mathbf{0}_n$ denote the $n$-by-$n$ identity matrix with ones on the main diagonal and zeros elsewhere and an n-by-n matrix of zeros respectively. Although the upper-right bloack of (51) is related to the current estimated state. To be specific, the bias error propagates through the iteration, but the InEKF retains its good consistency and convergence compared to common EKF.

In the extended Kalman filter, the abstract form of the measurement model can be written as:

$$\boldsymbol{y}(t_{n+1}) = h(\boldsymbol{x}(t_{n+1}), \boldsymbol{n}(t_{n+1})) \tag{53}$$

where $\boldsymbol{n}$ is the measurement noise, which is assumed a Gaussian distribution with zero mean and a covariance matrix of $\boldsymbol{N}$.

$$\boldsymbol{n} = \left( \left(\boldsymbol{n}_{\boldsymbol{R}_s^w}\right)^{\mathrm{T}} \left(\boldsymbol{n}_{\boldsymbol{v}_v^v}\right)^{\mathrm{T}} \right)^{\mathrm{T}} \tag{54}$$

$$\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}_{6\times 1}, \boldsymbol{N}) \tag{55}$$

For problem considered in this study, the data-driven attitude is modeled using a noise $\boldsymbol{n}_{\boldsymbol{R}_s^w}$

$$\boldsymbol{y}_{\boldsymbol{R}_s^w}(t_{n+1}) = \boldsymbol{R}_s^w(t_{n+1}) \left(\boldsymbol{n}_{\boldsymbol{R}_s^w}(t_{n+1})\right)_\times \tag{56}$$

while the data-driven velocity can be expressed by the filter state, as follows:

$$\boldsymbol{v}_v^v = \begin{pmatrix} v_{v,\mathrm{lat}}^v \\ v_{v,\mathrm{lon}}^v \\ v_{v,\mathrm{up}}^v \end{pmatrix} = \boldsymbol{R}_s^v \left( \boldsymbol{R}_w^s \boldsymbol{v}_s^w + \left(\boldsymbol{\omega}^s\right)_\times \boldsymbol{p}_v^s \right) \tag{57}$$

We combine the pseudo-measurement and DL forward velocity assuming that the velocity samples are mostly corrupted by additive white Gaussian noise, and deriving the measurement model:

$$\boldsymbol{y}_{\boldsymbol{v}_v^v}(t_{n+1}) = \begin{pmatrix} v_{\mathrm{lat}}^v(t_{n+1}) \\ v_{\mathrm{lon}}^v(t_{n+1}) \\ v_{\mathrm{up}}^v(t_{n+1}) \end{pmatrix} + \boldsymbol{n}_{\boldsymbol{v}_v^v}(t_{n+1}) \tag{58}$$

where $v_{lat}^v$ and $v_{up}^v$ denote the lateral and vertical velocities, respectively, which are both roughly equal zero in the practice. They are expressed in this paper as DDNHC. $v_{\mathrm{lon}}^v$ is the longitudinal velocity, given by the DL based estimator.

$$\tilde{\boldsymbol{y}}_{\boldsymbol{R}_w^s}(t_{n+1}) = \tilde{\boldsymbol{R}}_w^s(t_{n+1}) \tag{59}$$

$$\tilde{\boldsymbol{y}}_{\boldsymbol{v}_v^v}(t_{n+1}) = \begin{pmatrix} 0 & \tilde{v}_{\mathrm{lon}}^v(t_{n+1}) & 0 \end{pmatrix}^{\mathrm{T}} \tag{60}$$

Following the InEKF methodology, the updated estimated state can be derived as:

$$\boldsymbol{S}(t_{n+1}) = \boldsymbol{H}(t_{n+1}) \boldsymbol{P}\left(t_{n+1}^-\right) \left(\boldsymbol{H}(t_{n+1})\right)^{\mathrm{T}} + \boldsymbol{N}(t_{n+1}) \tag{61}$$

$$\boldsymbol{K}(t_{n+1}) = \boldsymbol{P}\left(t_{n+1}^-\right) \left(\boldsymbol{H}(t_{n+1})\right)^{\mathrm{T}} / \boldsymbol{S}(t_{n+1}) \tag{62}$$

$$\boldsymbol{e}\left(t_{n+1}^+\right) = \boldsymbol{K}(t_{n+1}) \boldsymbol{r}(t_{n+1}) \tag{63}$$

$$\hat{\boldsymbol{\chi}}_s^w\left(t_{n+1}^+\right) = \exp_{\mathbf{G}_{\mathrm{SE}_2(3)}}\left(\boldsymbol{\xi}_s^w\left(t_{n+1}^+\right)\right) \hat{\boldsymbol{\chi}}_s^w\left(t_{n+1}^-\right) \tag{64}$$

$$\hat{\boldsymbol{\delta}}_{\boldsymbol{\omega}_s^s}\left(t_{n+1}^+\right) = \hat{\boldsymbol{\delta}}_{\boldsymbol{\omega}_s^s}\left(t_{n+1}^-\right) + \boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}} \tag{65}$$

$$\hat{\boldsymbol{\delta}}_{\boldsymbol{f}_s^s}\left(t_{n+1}^+\right) = \hat{\boldsymbol{\delta}}_{\boldsymbol{f}_s^s}\left(t_{n+1}^-\right) + \boldsymbol{\xi}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}} \tag{66}$$

$$\hat{\boldsymbol{R}}_s^v\left(t_{n+1}^+\right) = \exp_{\mathbf{G}_{\mathrm{SO}(3)}}\left(\boldsymbol{\xi}_{\boldsymbol{R}_s^v}\right) \hat{\boldsymbol{R}}_s^v\left(t_{n+1}^-\right) \tag{67}$$

$$\hat{\boldsymbol{p}}_v^s\left(t_{n+1}^+\right) = \hat{\boldsymbol{p}}_v^s\left(t_{n+1}^-\right) + \boldsymbol{\xi}_{\boldsymbol{p}_v^s}\left(t_{n+1}^+\right) \tag{68}$$

$$\hat{\boldsymbol{P}}\left(t_{n+1}^+\right) = \left(\boldsymbol{I}_{21} - \boldsymbol{K}(t_{n+1}) \boldsymbol{H}(t_{n+1})\right) \hat{\boldsymbol{P}}\left(t_{n+1}^-\right) \tag{69}$$

To derive $\boldsymbol{H}$, which is the measurement Jacobian matrix with respect to the linearized error, first the measurement residual is calculated as:

$$\boldsymbol{r}_{\boldsymbol{R}_s^w}(t_{n+1}) = \log_{\mathbf{G}_{\mathrm{SO}(3)}}\left( \tilde{\boldsymbol{y}}_{\boldsymbol{R}_s^w}(t_{n+1}) \left(\hat{\boldsymbol{y}}_{\boldsymbol{R}_s^w}(t_{n+1})\right)^{\mathrm{T}} \right) \tag{70}$$

$$\boldsymbol{r}_{\boldsymbol{v}_v^v}(t_{n+1}) = \tilde{\boldsymbol{y}}_{\boldsymbol{v}_v^v}(t_{n+1}) - \hat{\boldsymbol{y}}_{\boldsymbol{v}_v^v}(t_{n+1}). \tag{71}$$

where $\boldsymbol{H}(t_{n+1})$ is the measurement Jacobian matrix with respect to linearized error (22) and thus given as:

$$\boldsymbol{H}^{\boldsymbol{R}_s^w} = \begin{pmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 \end{pmatrix} \tag{72}$$

$$\boldsymbol{H}^{\boldsymbol{v}_v^v} = \begin{pmatrix} \boldsymbol{H}_{\boldsymbol{R}_s^w}^{\boldsymbol{v}_v^v} & \boldsymbol{H}_{\boldsymbol{v}_s^w}^{\boldsymbol{v}_v^v} & \boldsymbol{H}_{\boldsymbol{p}_s^w}^{\boldsymbol{v}_v^v} & \boldsymbol{H}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}}^{\boldsymbol{v}_v^v} & \boldsymbol{H}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}}^{\boldsymbol{v}_v^v} & \boldsymbol{H}_{\boldsymbol{R}_s^v}^{\boldsymbol{v}_v^v} & \boldsymbol{H}_{\boldsymbol{p}_v^s}^{\boldsymbol{v}_v^v} \end{pmatrix} \tag{73}$$

$$\boldsymbol{H}_{\boldsymbol{R}_s^w}^{\boldsymbol{v}_v^v} = \boldsymbol{0}_3 \tag{74}$$

$$\boldsymbol{H}_{\boldsymbol{v}_s^w}^{\boldsymbol{v}_v^v} = \hat{\boldsymbol{R}}_s^v \left(\hat{\boldsymbol{R}}_s^w\right)^{\mathrm{T}} \tag{75}$$

$$\boldsymbol{H}_{\boldsymbol{p}_s^w}^{\boldsymbol{v}_v^v} = \boldsymbol{0}_3 \tag{76}$$

$$\boldsymbol{H}_{\boldsymbol{\delta}_{\boldsymbol{\omega}_s^s}}^{\boldsymbol{v}_v^v} = \hat{\boldsymbol{R}}_s^v \left(\hat{\boldsymbol{p}}_v^s\right)_\times \tag{77}$$

$$\boldsymbol{H}_{\boldsymbol{\delta}_{\boldsymbol{f}_s^s}}^{\boldsymbol{v}_v^v} = \boldsymbol{0}_3 \tag{78}$$

$$\boldsymbol{H}_{\boldsymbol{R}_s^v}^{\boldsymbol{v}_v^v} = -\left( \hat{\boldsymbol{R}}_s^v \left( \left(\hat{\boldsymbol{R}}_s^w\right)^{\mathrm{T}} \hat{\boldsymbol{v}}_s^w + \left(\boldsymbol{\omega}^s\right)_\times \hat{\boldsymbol{p}}_v^s \right) \right)_\times \tag{79}$$

$$\boldsymbol{H}_{\boldsymbol{p}_v^s}^{\boldsymbol{v}_v^v} = \hat{\boldsymbol{R}}_s^v \left(\boldsymbol{\omega}^s\right)_\times \tag{80}$$

## Experimental results

### Experimental setup

To assess the performance of the proposed method, the experiments were conducted in a surface parking lot to simulate an underground parking environment. The driving speed of the vehicle followed the parking lot's driving speed limit. The dataset contains 11 sequences for validating VDR algorithms The experimental sequences are shown in Fig. 5. A Novatel SPAN with an ISA-100C IMU was installed on the car to provide ground truth position at centimeter level accuracy, velocity and attitude solution at 200 Hz. A Huawei Mate 30 was chosen as the experimental platform. The IMU embedded in this smartphone was an STMicroelectronics LSM6DSM. The gyroscope rate noise density quoted by the manufacturer was $3.8 \times 10^{-3} \circ \mathrm{s}^{-1}\ \mathrm{Hz}^{-0.5}$, while during its acceleration the noise density became $g \times 90 \times 10^{-6} \mathrm{Hz}^{-0.5}$ ($g \approx 9.80\,\mathrm{m/s^2}$), with both values corresponding to the phone's high-performance mode. The NovAtel SPAN antenna was rigidly mounted on the vehicle roof, while the ISA-100C IMU and the smartphone were also rigidly mounted to a custom bracket at a fixed mounting angle, as shown in Fig. 4. Before data collection, the vehicle was driven to a location where the observation environment was good enough for optimal convergence. The data acquisition software was custom developed and its source


**Fig. 4** Data collection platform

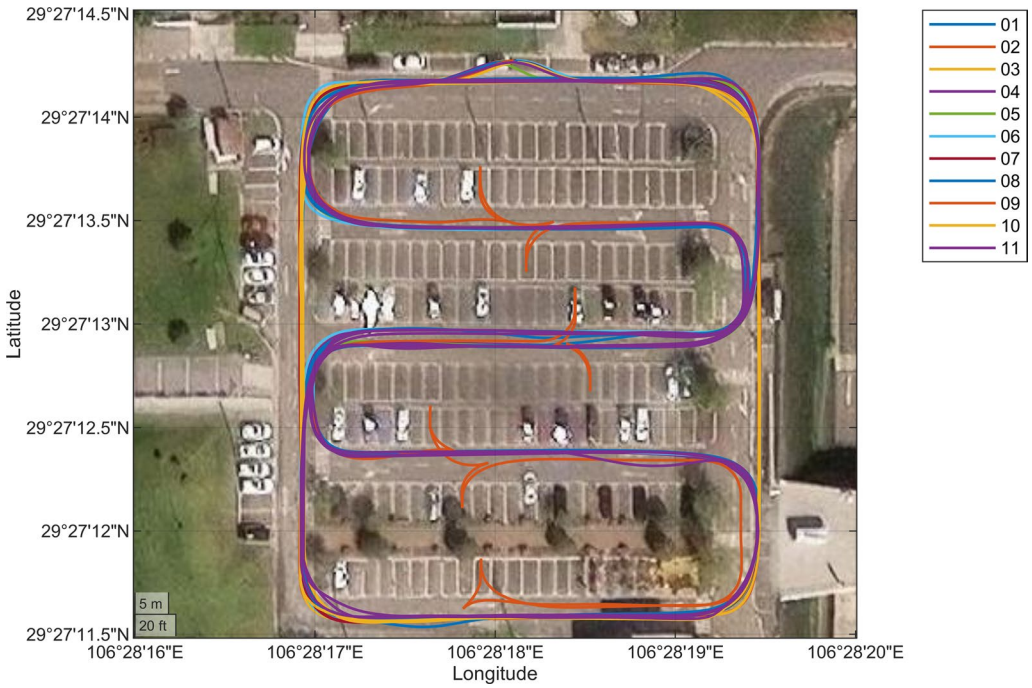


**Fig. 5** Experimental sequences

code has been published at: https://github.com/DragonEmperorG/VDRDataCollector. The software is based on the implementation of GNSSLogger by ISTA-UniBwM (Sharma et al., 2021). The parameter samplingPeriodUs in SensorManager was set to SENSOR_DELAY_FASTEST. In data processing, the raw sensor data were downsampled to 200 Hz.

#### Evaluation metrics

To assess the performance of the proposed modules, the evaluation metrics proposed in the KITTI benchmark suite were adopted (Geiger et al., 2012), but the evaluation sequence lengths were changed:

1. Relative rotational error ($E_{r_{\text{rel}}}$). This is the relative rotational increment error for $\mathcal{F}$ in degree per kilometer (81);
2. Relative translation error ($E_{t_{\text{rel}}}$): This is the averaged relative translation increment error for $\mathcal{F}$, as a percentage of the traveled distance (82);
3. Relative horizontal translation error ($E_{t_{\text{hor}}}$). This is the averaged relative horizontal translation increment error for $\mathcal{F}$, again as a percentage of the traveled distance (83).

$$E_{r_{\text{rel}}}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j)\in\mathcal{F}} \angle\left[\nabla\Delta\boldsymbol{\chi}_{\text{eva}}\right] \tag{81}$$

$$E_{t_{\text{rel}}}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j)\in\mathcal{F}} \left\|\nabla\Delta\boldsymbol{\chi}_{\text{eva}}\right\|_2 \tag{82}$$

$$E_{t_{\text{hor}}}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j)\in\mathcal{F}} \left\|\nabla\Delta\boldsymbol{\chi}_{\text{eva}}\right\|_{\text{hor}} \tag{83}$$

$$\nabla\Delta\boldsymbol{\chi}_{\text{eva}} = \left(\hat{\boldsymbol{\chi}}_{\text{eva},j} \ominus \hat{\boldsymbol{\chi}}_{\text{eva},i}\right) \ominus \left(\boldsymbol{\chi}_{\text{eva},j} \ominus \boldsymbol{\chi}_{\text{eva},i}\right) \tag{84}$$

where $\mathcal{F}$ is a set of frames $(i,j)$, which included all possible sub-sequences of length $(100, 200, \ldots, 900)$ meters; $\hat{\boldsymbol{\chi}}_{\text{eva}} \in SE(3)$ and $\boldsymbol{\chi}_{\text{eva}} \in SE(3)$ are the estimated and true smartphone poses, repectively; $\ominus$ denotes the inverse compositional operator; $\angle[\cdot]$ is the rotation angle; $\|\cdot\|_2$ is the translation; and $\|\cdot\|_{\text{hor}}$ is the horizontal translation.

The AI-IMU method formed the basis for this experimental process. Different combinations of data-driven measurement were used:

- AI-IMU (Brossard et al., 2020a): This is basically the AI-IMU method which has Two-Dimensional (2D) velocity constraints. We have modified the original open source code to match the dataset in this paper.
- DeepOdo (Wang et al., 2023a): This method is the same as the AI-IMU method with the addition of data-driven velocity. The DeepOdo method can be viewed as an InEKF with 3D velocity constraints. We reimplemented the DeepOdo method and integrated it through InEKF. Since only IMU data is used in this paper, the barometer data is not utilized as an input.
- DeepOri: This method is also similar to AI-IMU, only in this case data-driven attitude is added. The DeepOri method can be viewed as an InEKF with 2D velocity and 3D attitude constraints.
- Proposed: the proposed approach includes both data-driven attitude and velocity and the data-driven filter adapter which can be viewed as an InEKF with 3D velocity and 3D attitude constraints.

### Results

To test the framework using custom datasets, we followed the same protocol as in the AI-IMU method: First, the AVNet was trained without the evaluated sequence as explained in 4.2.3. Then, the adapter was trained without the evaluated sequence. Finally, the combined data- and model-driven method was applied to the test sequence and compared against the ground-truth.

#### *Trajectory results in car park scenarios*

The detailed statistic results are shown in Table 1 and some resulting trajectories are illustrated in Figs. 6, 7, 12, and 13, where there are two main trajectories with different driving modes, to simulate the underground parking situation.

**Rectangular trajectory results**

Rectangular trajectories were designed to simulate searching for parking spaces in a parking lot. Sequences 01, 02, 03, 07 and 10 belonged to the rectangular trajectory. For each trajectory, the vehicle started from one parking space and ended up in the same parking space. The length of one circle around the parking lot was about 285 m. For the basic rectangular trajectory, the AI-IMU method performed the worst, while the DeepOri and DeepODo methods performed better, and only the proposed method maintained good performance in all driving modes. The worse trajectory results mainly occurred at the starting and turning sections. Here, sequence 01 was taken to illustrate how the DDATT and DDODO constrains the starting section. Sequence 01 contained about 30 s of parking at the beginning of the sequence. We tried to perform dead reckoning from the moment when the vehicle started moving on sequence 01. The results were shown in Fig. 6 with solid line. For each

**Table 1** Results from different combinations of data- and model-driven methods

| Sequence number | Length (m) | Duration (s) | AI-IMU method | | | DeepOdo method | | | DeepOri method | | | Proposed method | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $E_{r_{rel}}$ ((°)/m) | $E_{t_{rel}}$ (%) | $E_{t_{hor}}$ (%) | $E_{r_{rel}}$ ((°)/m) | $E_{t_{rel}}$ (%) | $E_{t_{hor}}$ (%) | $E_{r_{rel}}$ ((°)/m) | $E_{t_{rel}}$ (%) | $E_{t_{hor}}$ (%) | $E_{r_{rel}}$ ((°)/m) | $E_{t_{rel}}$ (%) | $E_{t_{hor}}$ (%) |
| 01 | 282 | 124 | 52 | 15 | 15 | 16 | 5 | 4 | 12 | 6.10 | 5.50 | 4.3 | 2.06 | 0.42 |
| 02 | 285 | 122 | 50 | 25 | 25 | 35 | 9 | 9 | 12 | 1.91 | 1.39 | 2.1 | 1.23 | 0.43 |
| 03 | 285 | 208 | 85 | 66 | 66 | 62 | 18 | 18 | 56 | 8.49 | 8.44 | 1.4 | 0.93 | 0.42 |
| 04 | 520 | 168 | 80 | 11 | 11 | 160 | 13 | 13 | 34 | 6.34 | 5.17 | 2.6 | 2.82 | 0.39 |
| 05 | 525 | 216 | 124 | 25 | 25 | 164 | 9 | 8 | 8 | 2.45 | 1.68 | 2.0 | 1.44 | 0.33 |
| 06 | 524 | 370 | 288 | 1661 | 1633 | 131 | 22 | 22 | 20 | 3.53 | 3.43 | 1.2 | 0.84 | 0.29 |
| 07 | 844 | 316 | 131 | 21 | 21 | 126 | 9 | 9 | 8 | 2.59 | 1.42 | 1.2 | 1.95 | 0.34 |
| 08 | 1556 | 529 | 172 | 12 | 12 | 147 | 9 | 9 | 11 | 5.11 | 2.61 | 1.4 | 3.33 | 0.32 |
| 09 | 593 | 459 | 571 | 18272 | 6364 | 399 | 41 | 41 | 33 | 4.98 | 4.15 | 1.9 | 2.76 | 0.48 |
| 10 | 839 | 418 | 422 | 4767 | 4057 | 144 | 13 | 13 | 12 | 3.66 | 2.22 | 1.0 | 2.90 | 0.58 |
| 11 | 1553 | 807 | 386 | 174846 | 35661 | 255 | 26 | 26 | 26 | 7.61 | 5.80 | 1.1 | 3.40 | 0.45 |

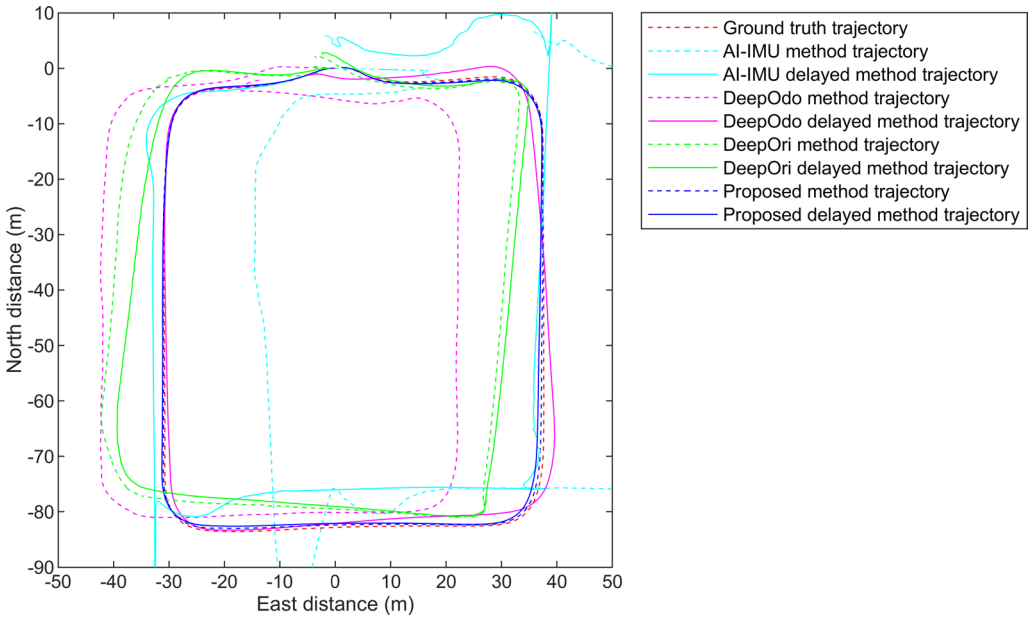


**Fig. 6** Trajectory results on sequence 01

compared methods, the dead reckoning results were all improved from the delayed start time. The AI-IMU method failed at the beginning of stationary sections. Due to the large errors of low-cost IMU, the DDHNC can not maintain the vehicle state during the stop periods. The cumulative error of the IMU inevitably manifested as the vehicle's velocity and position. Although the DeepOdo method can effectively constrain the velocity state, after several turns, the error of the attitude state became very large. The DeepOri method can effectively constrain the attitude state, but the positional errors accumulate along the trajectories. Since the framework did not have a separate module for determining vehicle motion, ZUPT and ZIHR updates were not executed in the first few seconds. The methods other than the proposed method accumulated large errors before the vehicle is in motion.

The comparison among sequences 01, 02 and 03 demonstrated the effect of the velocity factor on the approach. Sequences 01, 02 and 03 corresponded to traveling speeds of 15 km/h, 10 km/h and 5 km/h, respectively. As the speed decreased, the performance of the proposed method remained consistently well, while the

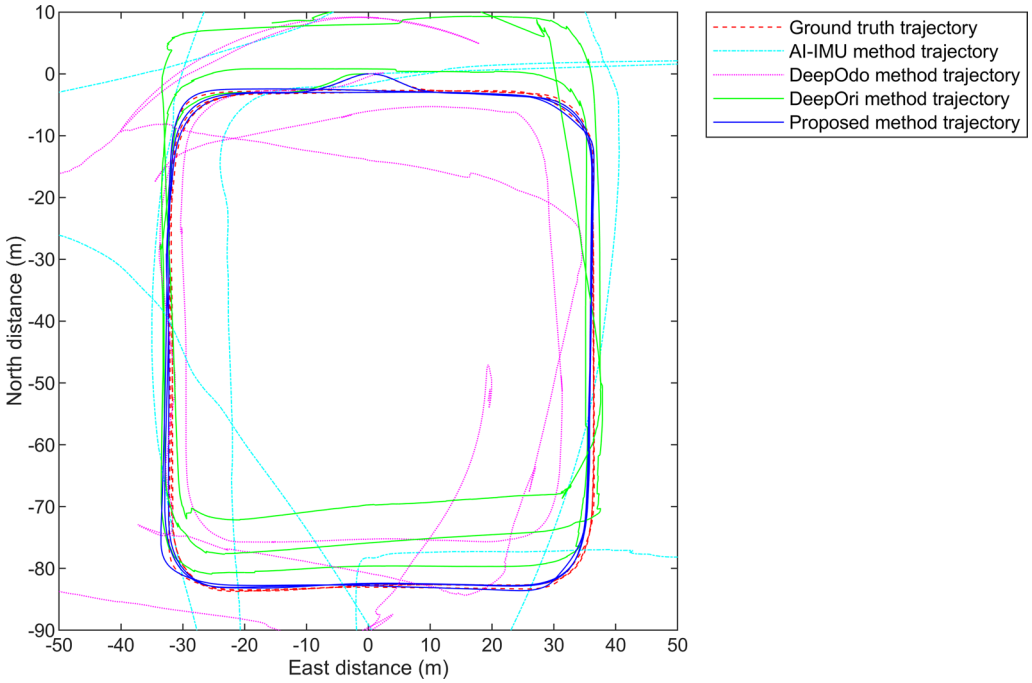


**Fig. 7** Trajectory results on sequence 10

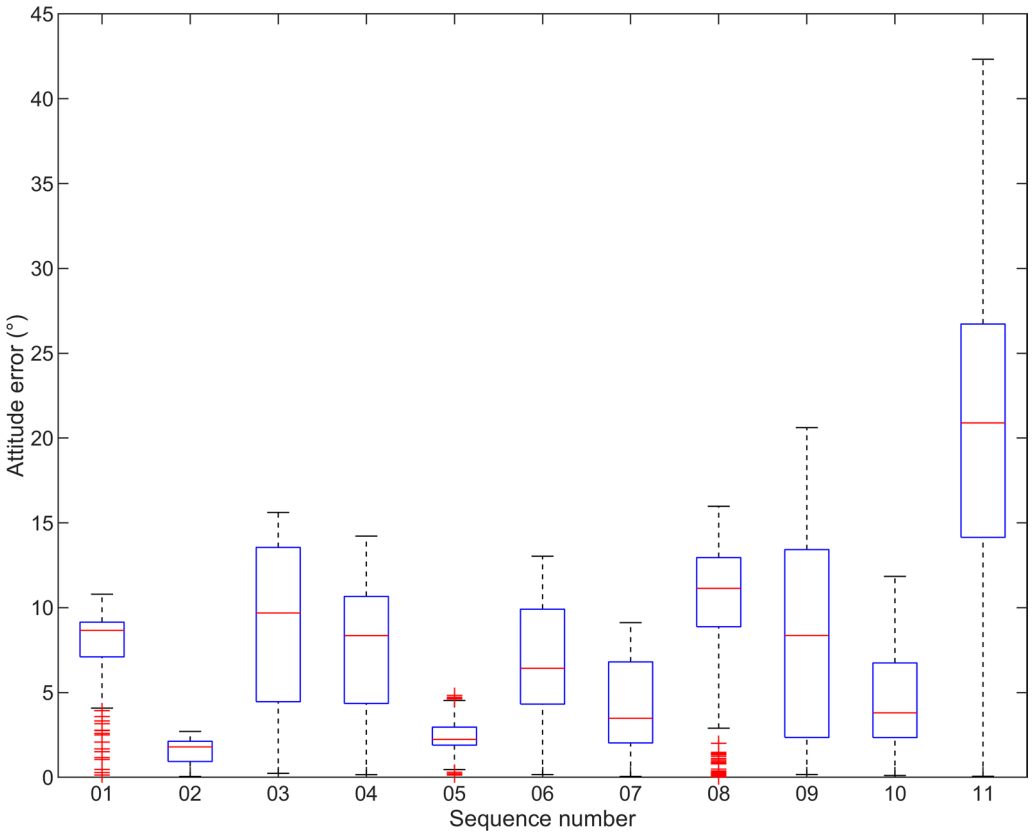


**Fig. 8** Errors of the data-driven attitude

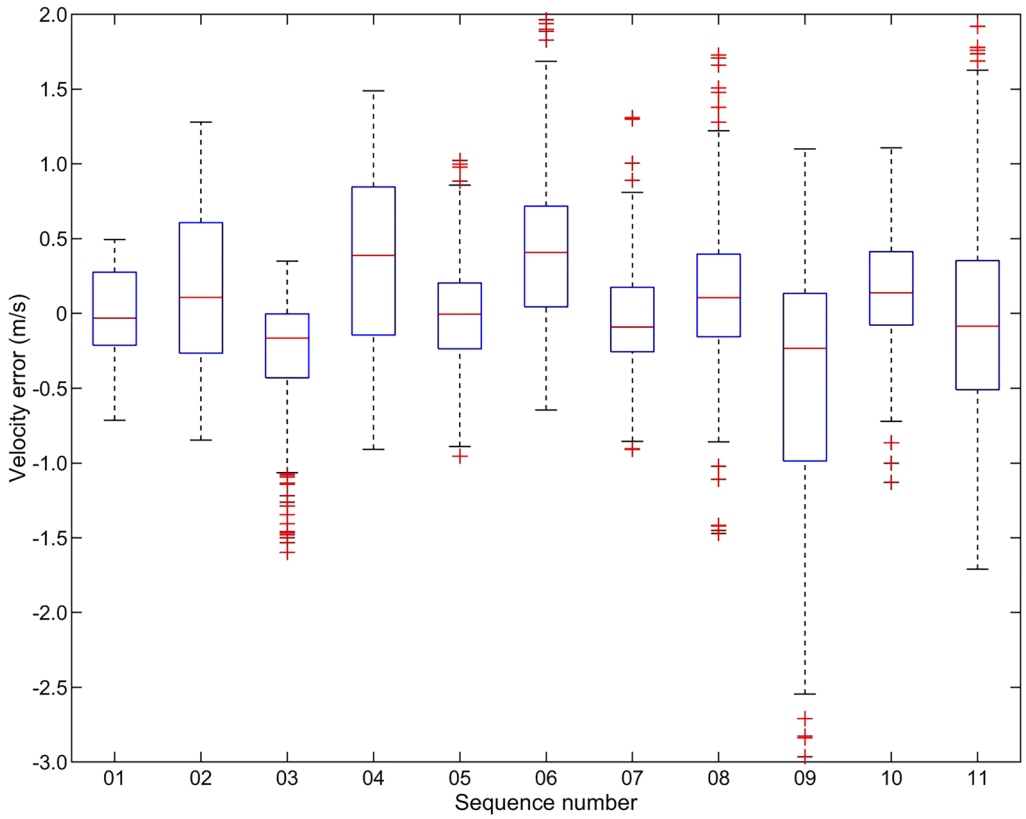


**Fig. 9** Errors of the data-driven velocity

compared methods showed a significant degradation. As can be seen from the Table 1, for the DeepOri method, sequences 02 achieved better results than sequence 01 and 03. This anomaly can be explained by the fact that the errors of DDATT is the smallest among the three sequences as seen in Fig. 8. Also, the DeepOri methods generally yielded the results with lower errors than the DeepOdo method. Gyroscope errors were the dominant source of dead reckoning errors compared with accelerometer errors. The DDATT measurements effectively limited gyroscope error accumulation which led to better performance.

Sequences 07 and 10 are designed to simulate longer and more realistic driving scenarios. Sequence 07 contains three loops of the rectangular trajectories without constant speed limit except for the maximum speed of 15 km/h, while sequence 10 is the same as sequence 07 with additional random stops. Both sequences 07 and 10 started at the origin and went on in a counterclockwise direction for 3 revolutions before returning to its origin. Since the AI-IMU method diverges so quickly, the Fig. 7 only shows trajectories near the ground truth. The AI-IMU (cyan dashed line) cannot see the shape of the trajectory. The DeepOdo method (magenta dashed-dotted line) has a deviation in heading at the first 90 ° turn. The DeepOri method (green dotted line) can see the shape of the trajectory, but the trajectory has large deviations. Only the proposed method (blue solid line) followed the ground truth (red dashed line) trajectory for the entire sequence. (0.6 km, 8 min). Comparing the evaluation metrics of sequences 07 and 10, both DDATT and DDODO constraints can achieve some constraints on the random stop scenario. And the proposed method achieved minimal difference in results when dealing with these two trajectories. It meant that the random stops factor did not have a significant impact on the performance of the proposed method.

**S-shaped trajectory results**

To further validate the effect of 6D geometric constraints on the results, S-shaped Trajectories were designed to cover more complex driving scenarios. Sequences 04, 05, 06, 08 and 11 belonged to the S-shaped trajectory. For S-shaped trajectories, the experiments were designed with the same influences as for the rectangular trajectories. Sequences 04, 05 and 06 corresponded to traveling speeds of 15 km/h, 10 km/h and 5 km/h, respectively. The effect of the

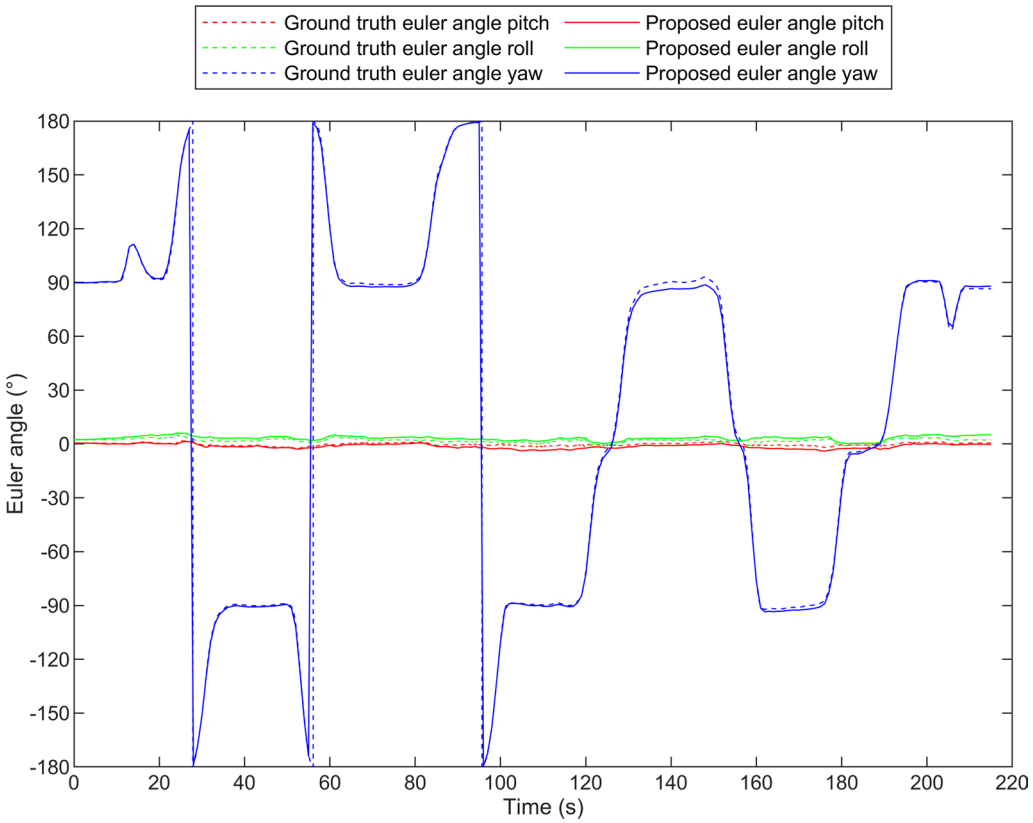


**Fig. 10** Data-driven attitude result on sequence 05

velocity factor on accuracy showed a pattern similar to that of the rectangular trajectory. As can be seen in Table 1, the results for sequence 05 are slightly better than those for sequence 04 and 06. This is due to the fact that the accuracy of the DDATT and DDODO measurements for sequence 05 are slightly better than the other two which can be seen in Figs. 8 and 9. The estimated DDATT and DDODO measurements by AVNet is demonstrated in Figs. 10 and 11.

In Fig. 8, the detailed comparison of the attitude errors of the sequences is presented by the box plot. The red line in the middle the box represents the 50th percentile, and the top and bottom of the box represent the 75th and 25th percentiles, respectively. Fig. 8 shows that the rotational errors mainly fall within 10 °. The mean rotational error of the whole dataset is 10 °. The minimum error of 3 ° is achieved in sequence 02., while the median error of sequence 11 is 21 °. The driving velocity and trajectory do not have a significant effect on the predicted measurement, but the duration does affect the cumulative results.

The detailed comparison of the forward velocity errors of the sequences is shown in Fig. 9 using box plots. It is evident that the velocity errors mainly remain within 1 m/s. The median error of all sequences fell within $(-0.5, 0.5)$ m/s. The mean absolute velocity error of the whole dataset is 0.5 m/s. The complex turning situations can have an impact on the estimate. The DDODO measurement with the above accuracy might serve as effective constraints for VDR.

For sequence 08, we similarly tested three loops of the S-shaped trajectories to achieve longer distances with the same speed limit as sequence 07. Random stops were also incorporated in sequence 11 in Fig. 12. Sequence 11 initiated from the origin with westward orientation and continued for 3 revolutions before returning to its origin. It was the most challenging trajectory of the S-shaped trajectory experiment group. The proposed method (blue solid line) followed the ground truth (red dashed line) trajectory for the entire sequence. The AI-IMU method can barely maintain the three loops of the rectangular trajectories. The DeepOdo method failed at the second turn while the DeepOri method failed at the fourth turn in sequence 11. The attitude constraints might be more important in this low-speed scenarios. Only the proposed method can follow the whole trajectory constituting the most rigorous experimental scenario. Although attitude constraints are more important, it is necessary to impose both attitude and velocity constraints.

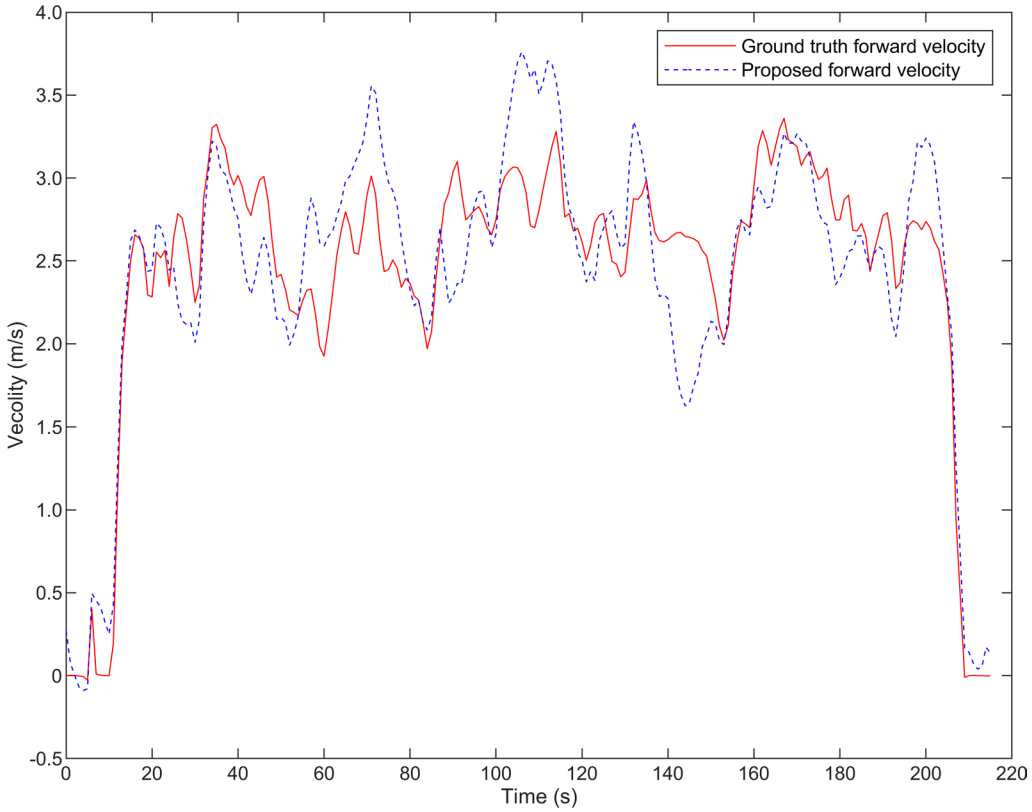


**Fig. 11** Data-driven velocity result on sequence 05

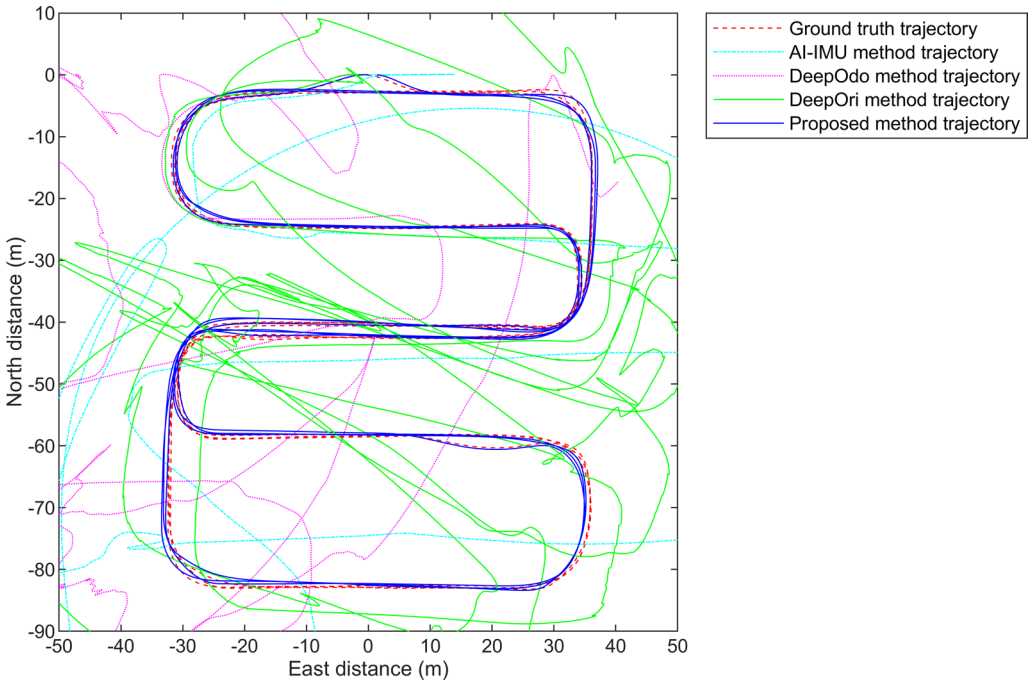


**Fig. 12** Trajectory results on sequence 11

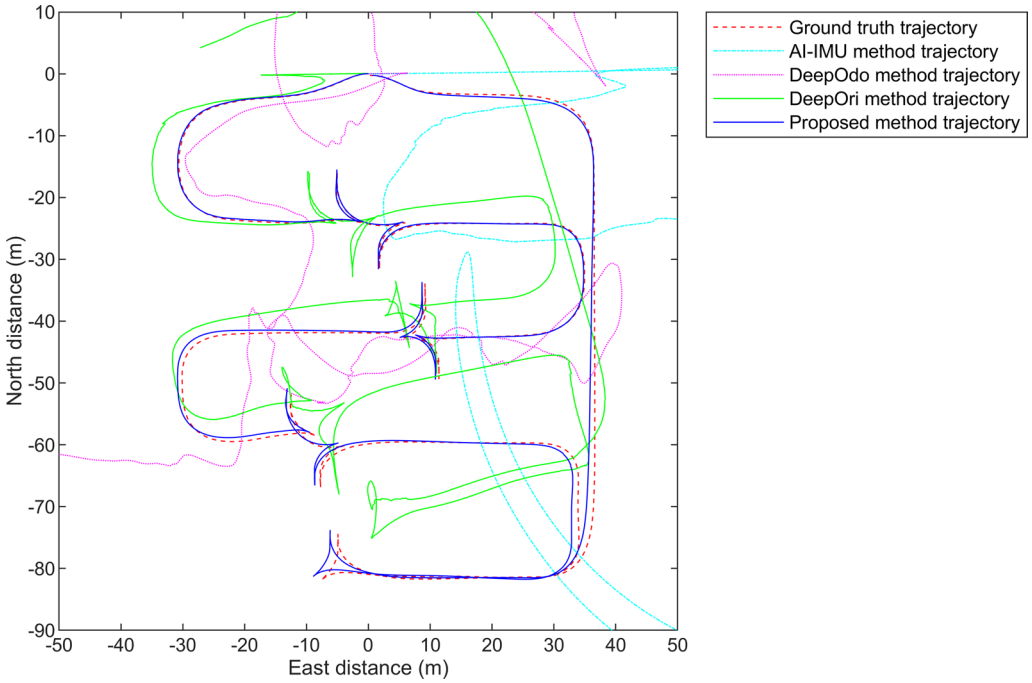


**Fig. 13** Trajectory results on sequence 09

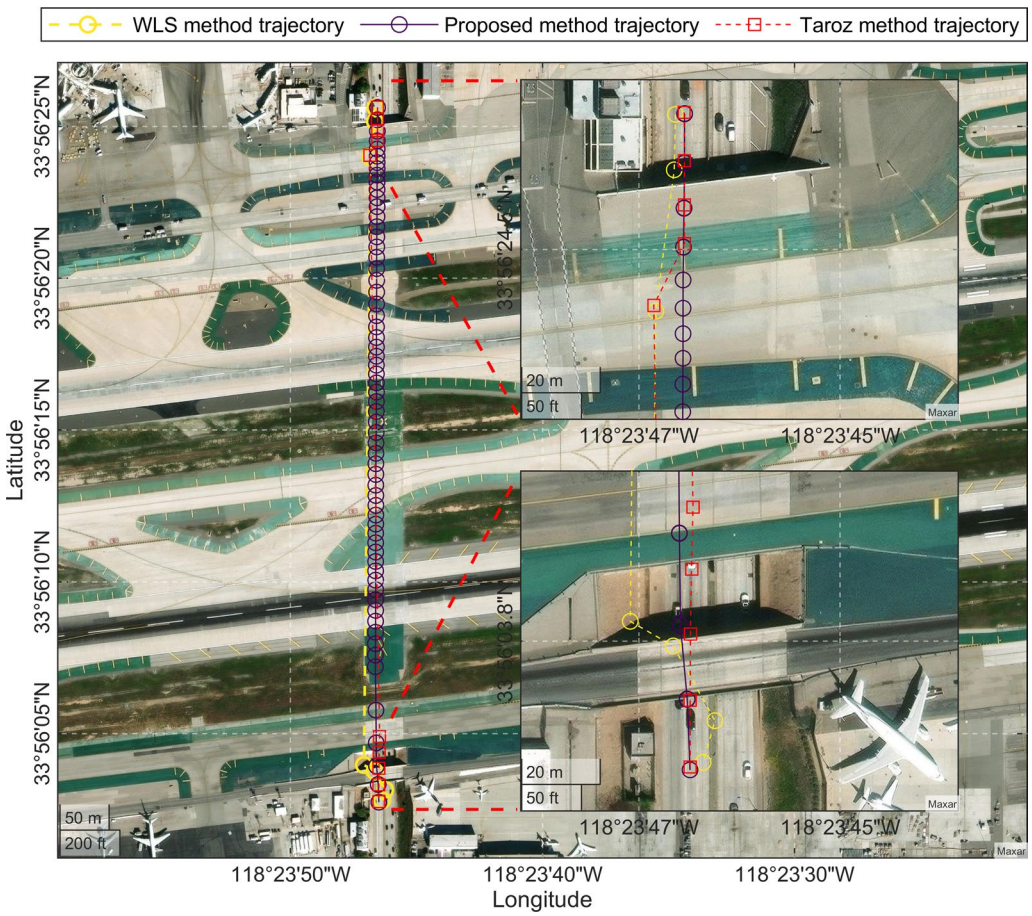


**Fig. 14** The selected section of trajectory visualization

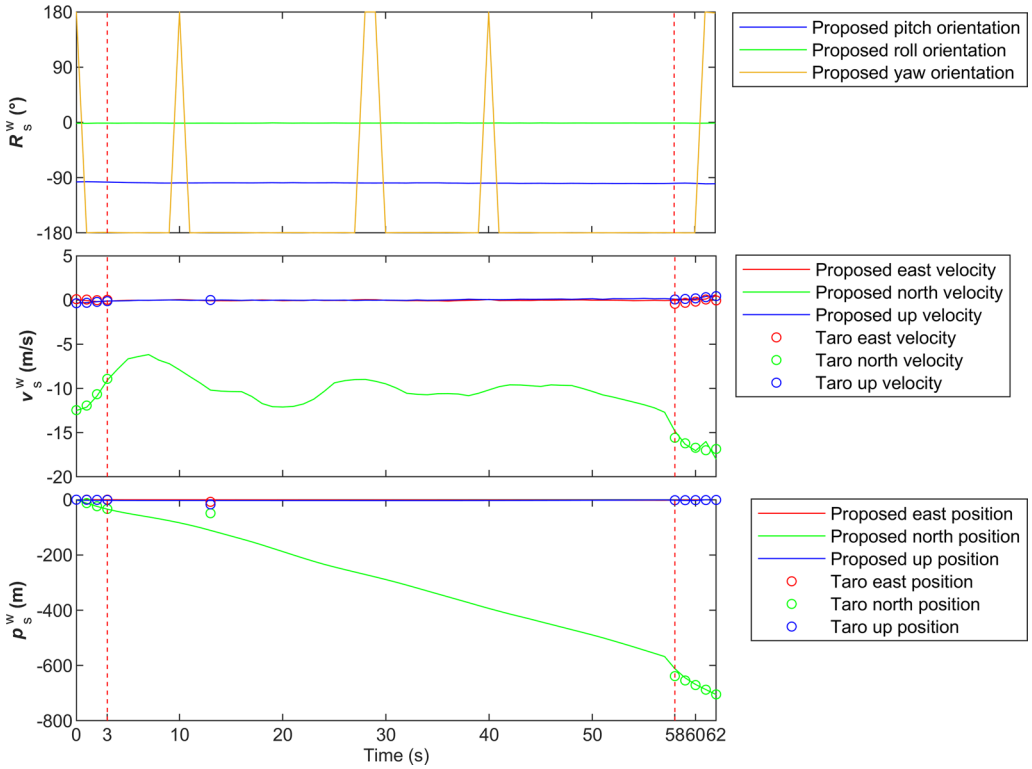


**Fig. 15** Main filter states of the selected section

Sequence 09 included reverse parking to simulate a more realistic situation in Fig. 13. This trajectory started at the origin heading west and included random reverse parking in 7 spaces during the travel. At the first stopping point, the position error of the DeepOri method is approximately 5 m while the estimated position of the proposed method is almost identical to the ground truth. Owing to data-driven measurements, the proposed method exhibited no significant performance deviation compared to other trajectories. The proposed method (blue solid line) followed the ground truth (red dashed line) trajectory for the entire sequence (0.6 km, 8 min). The results demonstrate the method's superior performance relative to other methods with partial data-driven measurements integration. With DDATT, $E_{t_{\text{hor}}}$ can be constrained within 10 % within the datasets. With all the data-driven measurements, $E_{t_{\text{hor}}}$ can be constrained within 1 %. The proposed method can cover reverse driving scenarios. However, the frequent turning of the vehicle definitely contributed to the accelerated accumulation of positional error.

### *A case study in tunnel scenarios*

Existing public datasets for VDR research are usually built on automotive-grade sensors (Geiger et al., 2013), while smartphone-based inertial positioning datasets are primarily designed for pedestrian navigation (Chen & Pan, 2024). Few datasets can provide synchronized smartphone sensors data with high-frequency ground-truth vehicle motion data, especially in challenging environments. To improve high precision GNSS positioning and navigation accuracy on smartphones, Google has hosted a series of competitions called Google Smartphone Decimeter Challenge (GSDC) since 2021 and released datasets used in those competitions (Suzuki, 2023). Although the purpose of GSDC is primarily to enhance research in smartphone GNSS signal processing, there exist the sections of trajectories where the received GNSS signals are severely degraded (e.g., due to multipath interference, urban canyon effects, or tunnel obstruction) or even completely unavailable, leading to positioning discontinuities and cumulative drift errors. A section of trajectories from the GSDC 2023-2024 was selected to evaluate the positioning accuracy of the proposed method (Chow et al., 2023). Fig. 14 shows the qualitative visualizations of the tracking results. The selected section came from 2022-02-23-22-35-us-ca-lax-m\pixel5 trajectory in the GSDC 2023-2024 test datasets that was in a tunnel. The entire length of the tunnel was about 578 m, and the duration of the GNSS signal interruption was about 55 s. In order to facilitate the presentation of the selected section, the trajectory data between

1645656471438 ms and 1645656533438 ms was intercepted which means that the 0 s in Fig. 15 corresponded to 1645656471438 ms. The last GNSS observation timestamp in unix time milliseconds before entering the tunnel was 1645656474438 ms or 3 s while the first timestamp after leaving the tunnel was 1645656529438 ms or 58 s. The following will be expressed in terms of local intercepted time. Although the GNSS data was missing in the period between 3 s and 58 s, the vehicle might actually be in a situation where the GNSS signal was blocked by the tunnel from 2 s to 60 s. The DMDVDR method was applied to the period from 2 s to 60 s. The training datasets of GSDC 2023-204 had 39 trajectories collected by Pixel5, which were used to retrain the AVNet model as the data-driven part of the proposed method. The initial state of the filter was considered convergent by integration of IMU and GNSS. The results obtained by the open-source factor graph optimization method proposed by Suzuki (2024) who was the second place winner of the competition were used as ground truth. The results of Weighted-Least-Squares (WLS)-based processing of the pseudo-ranges observed by the smartphone were plotted as a baseline. As can be seen in the enlarged view in the upper right corner of Fig. 14, both Taroz's and proposed's methods can smoothly estimate the position when entering the tunnel. The WLS method failed in the tunnel environment, and the Taroz method also was affected. Although GNSS signals may be received near the tunnel entrances and exits, the number of visible satellites is low and the signal quality is poor which make GNSS observations less credible. The final absolute position error of the proposed method relative to the Taroz method at 60 s was [−2.5 m, 2.2 m, −1.5 m] in ENU coordinate. The proposed method achieved 0.64 % final translation drift in the tunnel. The core state values of the InEKF are shown in Fig. 15. With the constraints of DDATT, DDODO and DDNHC, the vehicle's attitude and velocity did not diverge away rapidly.

## Discussion

The first attempt in this paper was to apply the AI-IMU to smartphones. From the results of AI-IMU, the NHC constraint can not suppress sensor dispersion well. It might be because significant differences in IMU accuracy between smartphones and RT3003 makes the InEKF not as effective as it should be. Numerous sub-sequences which have different initial states and duration were tried to locate the cause of AI-IMU failure. The AI-IMU showed the expected effect when the initial phase of the trajectory did not contain long stationary periods. If significant filter state errors accumulated during stationary phases, the trajectory results were bound to be poorer. Although the sequence 07 result showed the AI-IMU's robustness when the car stops (Brossard et al., 2020a), more constraints needed to be imposed when it is applied to smartphones. The tunnel test results showed that forward velocity constraints were critical for horizontal error mitigation (Wang et al., 2023a). The NHC with ODO did work well on that situation, but there were still problems in the low-speed car park scenarios. The 3D velocity constraint limited the vehicle position to the current. However, if the vehicle's attitude was not restricted, the attitude state of the vehicle would diverge in filtering iterations due to gyroscope errors which means the vehicle's heading keeps changing. So, it was important to apply ZUPT and ZIHR at the same time when vehicles were at a standstill, otherwise the vehicle's heading would deviate considerably when resuming motion. The empirical threshold of the standard deviation of IMU was used to detect motionless state (Wang et al., 2023a). The data-driven attitude and velocity measurements allowed the empirical thresholds to be fused into the filters. In ideal conditions, these two observations were both fixed when the car is stationary. The attitude and velocity constraints can be seen as broader versions of the ZUPT and ZIHR which not only work in motionless states but also in motional states. Comparative analysis between DeepOdo and DeepOri revealed that the gyroscope error had a more pronounced effect compared to the accelerometer error. It should be mentioned that the experimental parking lot was slightly sloped due to the topography of the parking lot, which slightly complicated this particular dead reckoning problem. In addition, the vertical transition error was larger than the horizontal one, which may be due to the fixed gravity assumption.

Finally, in terms of real-time capability of the method, the real-time performance is considered in the design and implementation of the method. The iteration of Kalman filter accounts for the main computational cost (Brossard et al., 2020a). Since Kalman filters have been widely used in industries that require real-time capability and computational efficiency, there shall be no major problems for real-time application. For the deep neural network, the performance of the DeepOdo network was validated (Wang et al., 2023a), and the AVNet does not increase the number of parameters compared with DeepOdo. Hence the data-driven measurement can be integrated into the framework in real-time. The DMDVDR framework is not restricted to inertial dead reckoning of vehicles. The InEKF can easily be coupled with GNSS measurements which can achieve more robust and seamless indoor/outdoor positioning results.

## Conclusion

In this paper, a novel approach for inertial-only VDR was proposed using smartphones. The framework synergistically integrates data-driven measurements with model-driven filtering. The proposed AVNet learns the attitude and velocity features from raw IMU measurements, thus constraining the dead-reckoning process. Experiments were conducted in a parking and tunnel environment to demonstrate the proposed method's effectiveness and robustness.

From the results obtained, the following conclusions can be drawn:

1. By constraining both the attitude and velocity features of vehicles, it is possible to obtain accurate horizontal positional results using only a smartphone's IMU. The relative horizontal translation error can be less than 1 % in car park scenarios.
2. The AVNet is able to learn not only velocity features but also attitude features. It can be achieved that the mean rotational errors of the DDATT are less than 10 °, and the mean velocity errors are less than 1 m/s in low-speed scenarios. The integrated pseudo-measurements can correct system state in the updating process of an InEKF.
3. The InEKF demonstrated robust performance with low-cost IMU on smartphones.

In summary, the DMDVDR establishes a new paradigm for AI-enhanced inertial navigation. The current shortcomings of our approach are that the smartphone is fixed in collecting the data, and the dataset is relatively small. Future works involve generalizing the method to adapt to the capabilities of different smartphones and examine more complex driving scenarios.

**Abbreviations**

| | |
|---|---|
| **2D** | Two-dimensional |
| **6D** | Six-dimensional |
| **BiLSTM** | Bidirectional long short-term memory |
| **BLE** | Bluetooth low energy |
| **CNN** | Convolutional neural network |
| **DDATT** | Data-driven attitude |
| **DDNHC** | Data-driven non-holonomic constraint |
| **DDODO** | Data-driven odometry |
| **DL** | Deep learning |
| **DMDVDR** | Data and model driven vehicle dead reckoning |
| **DNN** | Deep neural network |
| **GNSS** | Global navigation satellite system |
| **GRU** | Gated recurrent unit |
| **GSDC** | Google smartphone decimeter challenge |
| **IMU** | Inertial measurement unit |
| **InEKF** | Invariant extended Kalman filter |
| **INS** | Inertial navigation system |
| **LSTM** | Long short-term memory |
| **NHC** | Non-holonomic constraint |
| **ODO** | Odometry |
| **ReLU** | Rectified linear unit |
| **VDR** | Vehicle dead reckoning |
| **WLS** | Weighted least squares |
| **ZIHR** | Zero integrated heading rate |
| **ZUPT** | Zero velocity update |

**Acknowledgements**

The authors would like to thank SICHUAN JIUZHOU AIR TRAFFIC CONTROL TECHNOLOGY CO., LTD for providing equipment and personnel support for the experiment, and thank Ministry of Education's Organized Scientific Research for providing grants.

**Author Contributions**

LQ and RZC proposed the initial idea of this work; LQ, XGN and LLL designed the experiments; LQ and QHH carried out datasets collection and data curation; LQ carried out the algorithm implementation; XCL and ZXW helped with the result analysis and discussions; XGN, LLL, GYG, and RZC contributed to constructive guidance and advice; LQ wrote the original manuscript; LQ, XCL, XGN, GYG, ZXW and RZC assisted in manuscript revision. All authors have approved the submitted version.

**Funding**

This study was supported by the National Key Research and Development Program of China (grant nos. 2023YFB3906600), and the NSFC (grant no. 42201460).

**Availability of data and materials**

The datasets generated and analyzed in the current study are available from the corresponding author upon reasonable request.

## Declarations

**Competing interests**

Ruizhi Chen is an editorial board member for Satellite Navigation and was not involved in the editorial review or decision to publish this article. All authors declare that they have no competing interests.

Received: 7 December 2024 Revised: 20 May 2025 Accepted: 21 May 2025
Published online: 20 June 2025

**References**

Bai, S., Wen, W., Yu, Y., & Hsu, L.-T. (2024). Invariant extended kalman filtering for pedestrian deep-inertial odometry. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLVIII–4–2024*, 607–612. https://doi.org/10.5194/isprs-archives-XLVIII-4-2024-607-2024

Barrau, A., & Bonnabel, S. (2017). The invariant extended kalman filter as a stable observer. *IEEE Transactions on Automatic Control, 62*, 1797–1812. https://doi.org/10.1109/TAC.2016.2594085

Barrau, A., & Bonnabel, S. (2023). The geometry of navigation problems. *IEEE Transactions on Automatic Control, 68*, 689–704. https://doi.org/10.1109/TAC.2022.3144328

Brossard, M., Barrau, A., & Bonnabel, S. (2019). RINS-W: Robust Inertial Navigation System on Wheels. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2068–2075. https://doi.org/10.1109/IROS40897.2019.8968593. https://ieeexplore.ieee.org/document/8968593/

Brossard, M., Barrau, A., & Bonnabel, S. (2020a). Ai-imu dead-reckoning. *IEEE Transactions on Intelligent Vehicles, 5*, 585–595. https://doi.org/10.1109/TIV.2020.2980758

Brossard, M., Bonnabel, S., & Barrau, A. (2020b). Denoising IMU Gyroscopes with Deep Learning for Open-Loop Attitude Estimation. *IEEE Robotics and Automation Letters, 5*(3), 4796–4803. https://doi.org/10.1109/LRA.2020.3003256. arXiv:2002.10718.

Brossard, M., Barrau, A., Chauchat, P., & Bonnabel, S. (2022). Associating uncertainty to extended poses for on lie group imu preintegration with rotating earth. *IEEE Transactions on Robotics, 38*, 998–1015. https://doi.org/10.1109/TRO.2021.3100156

Callebaut, G., Ottoy, G., & Strycker, L.D. (2019). Bring your own sensor: Use your android smartphone as a sensing platform. In *2019 IEEE Sensors Applications Symposium (SAS)*, pp. 1–5. https://doi.org/10.1109/SAS.2019.8705987. https://ieeexplore.ieee.org/document/8705987/

Chen, C., & Pan, X. (2024). Deep Learning for Inertial Positioning: A Survey. *IEEE Transactions on Intelligent Transportation Systems, 25*, 10506–10523. https://doi.org/10.1109/TITS.2024.3381161. arXiv:2303.03757.

Chen, C., Lu, C. X., Wahlstrom, J., Markham, A., & Trigoni, N. (2021). Deep Neural Network Based Inertial Odometry Using Low-Cost Inertial Measurement Units. *IEEE Transactions on Mobile Computing, 20*(4), 1351–1364. https://doi.org/10.1109/TMC.2019.2960780

Chow, A., Orendorff, D., Fu, M., Khider, M., Dane, S., & Gulati, V. (2023). Google Smartphone Decimeter Challenge 2023-2024. https://kaggle.com/competitions/smartphone-decimeter-2023. Kaggle

Esfahani, M. A., Wang, H., Wu, K., & Yuan, S. (2020). OriNet: Robust 3-D Orientation Estimation with a Single Particular IMU. *IEEE Robotics and Automation Letters, 5*(2), 399–406. https://doi.org/10.1109/LRA.2019.2959507

Gao, R., Xiao, X., Zhu, S., Xing, W., Li, C., Liu, L., Ma, L., & Chai, H. (2021). Glow in the dark: Smartphone inertial odometry for vehicle tracking in gps blocked environments. *IEEE Internet of Things Journal, 8*, 12955–12967. https://doi.org/10.1109/JIOT.2021.3064342

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 3354–3361. https://doi.org/10.1109/CVPR.2012.6248074

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research, 32*(11), 1231–1237. https://doi.org/10.1177/0278364913491297

Guo, F., Yang, H., Wu, X., Dong, H., Wu, Q., & Li, Z. (2023). Model-based deep learning for low-cost imu dead reckoning of wheeled mobile robot. *IEEE Transactions on Industrial Electronics*, *71*(7), 7531–7541. https://doi.org/10.1109/TIE.2023.3301531

Herath, S., Yan, H., & Furukawa, Y. (2020). Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 3146–3152. https://doi.org/10.1109/ICRA40945.2020.9196860. https://ieeexplore.ieee.org/document/9196860/

Li, L., Huang, Q., Xu, K., Guo, G., & Chen, R. (2022). Vehicle positioning in underground space using a smart phone. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLVI–3/W1–2022*, 81–87. https://doi.org/10.5194/isprs-archives-XLVI-3-W1-2022-81-2022

Li, Z., Chen, R., Guo, G., Ye, F., Qian, L., Xu, S., Huang, L., & Chen, L. (2023). Dual-step acoustic chirp signals detection using pervasive smartphones in multipath and nlos indoor environments. *IEEE Internet of Things Journal*, *11*(4), 6494–6507 https://doi.org/10.1109/JIOT.2023.3312853

Liu, K., Jin, F., Hu, J., Xie, R., Gu, F., Guo, S., & Luo, J. (2022). Towards robust wifi fingerprint-based vehicle tracking in dynamic indoor parking environments: An online learning framework. *IEEE Transactions on Mobile Computing, 22*, 1–15. https://doi.org/10.1109/TMC.2022.3200411

Niu, X., Nassar, S., & El-Sheimy, N. (2007). An accurate land-vehicle mems imu/gps navigation system using 3d auxiliary velocity updates. *Navigation, Journal of the Institute of Navigation, 54*, 177–188. https://doi.org/10.1002/j.2161-4296.2007.tb00403.x

Niu, X., Peng, Y., Dai, Y., Chen, Q., Guo, C., & Zhang, Q. (2023). Camera-based lane-aided multi-information integration for land vehicle navigation. *IEEE/ASME Transactions on Mechatronics, 28*, 152–163. https://doi.org/10.1109/TMECH.2022.3192985

Niu, X., Ding, L., Wang, Y., & Kuang, J. (2024). Mgins: A lane-level localization system for challenging urban environments using magnetic field matching/gnss/ins fusion. *IEEE Transactions on Intelligent Transportation Systems*, *25*(10),14890–14904 https://doi.org/10.1109/TITS.2024.3386568

Sharma, H., Bochkati, M., & Pany, T. (2021). Time-synchronized gnss/imu data logging from android smartphone and its influence on the positioning accuracy. In: *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pp. 2000–2011. https://doi.org/10.33012/2021.17996. https://www.ion.org/publications/abstract.cfm?articleID=17996

Solà, J. (2017). Quaternion kinematics for the error-state Kalman filter. arXiv:1711.02508.

Suzuki, T. (2023). Precise position estimation using smartphone raw gnss data based on two-step optimization. *Sensors, 23*, 1205. https://doi.org/10.3390/s23031205

Suzuki, T. (2024). Second place winner of the smartphone decimeter challenge: An open-source factor graph optimization package for gnss and imu integration in smartphones. In *Proceedings of the 37th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2024)*, pp. 2703–2713. https://doi.org/10.33012/2024.19923. https://www.ion.org/publications/abstract.cfm?articleID=19923

Tang, H., Niu, X., Zhang, T., Li, Y., & Liu, J. (2022). OdoNet: Untethered Speed Aiding for Vehicle Navigation Without Hardware Wheeled Odometer. *IEEE Sensors Journal, 22*(12), 12197–12208. https://doi.org/10.1109/JSEN.2022.3169549. arXiv:2109.03091.

Wang, J., Weng, D., Qu, X., Ding, W., & Chen, W. (2023a). A Novel Deep Odometry Network for Vehicle Positioning Based on Smartphone. *IEEE Transactions on Instrumentation and Measurement, 72*, 1–12. https://doi.org/10.1109/TIM.2023.3240227

Wang, X., Zhuang, Y., Cao, X., Li, Q., Wang, Z., Cao, Y., & Chen, R. (2023b). SdoNet: Speed Odometry Network and Noise Adapter for Vehicle Integrated Navigation. *IEEE Internet of Things Journal, 10*(21), 19328–19343. https://doi.org/10.1109/JIOT.2023.3294947

Wang, Y., Cheng, H., & Meng, M.Q.-H. (2020). Pedestrian motion tracking by using inertial sensors on the smartphone. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4426–4431. https://doi.org/10.1109/IROS45743.2020.9341173. https://ieeexplore.ieee.org/document/9341173/

Wang, Y., Cheng, H., & Meng, M.Q.-H. (2022). Inertial odometry using hybrid neural network with temporal attention for pedestrian localization. *IEEE Transactions on Instrumentation and Measurement, 71*, 1–10. https://doi.org/10.1109/TIM.2022.3186704

Wang, Z., Wu, Y., & Niu, Q. (2020). Multi-sensor fusion in automated driving: A survey. *IEEE Access, 8*, 2847–2868. https://doi.org/10.1109/ACCESS.2019.2962554

Xu, K., Li, L., Chen, R., Li, Y., & Huang, Q. (2021). Application of smartphone based land vehicle navigation. In *2021 5th CAA International Conference on Vehicular Control and Intelligence, CVCI* 2021. https://doi.org/10.1109/CVCI54083.2021.9661130

Zhang, Q., Hu, Y., & Niu, X. (2020). Required lever arm accuracy of non-holonomic constraint for land vehicle navigation. *IEEE Transactions on Vehicular Technology, 69*(8), 8305–8316. https://doi.org/10.1109/TVT.2020.2995076

Zhang, Q., Lin, H., Ding, L., Chen, Q., Zhang, T., & Niu, X. (2024). Ransac-based fault detection and exclusion algorithm for single-difference tightly coupled gnss/ins integration. *IEEE Transactions on Intelligent Vehicles, 9*, 3986–3997. https://doi.org/10.1109/TIV.2023.3342274

Zhao, X., Deng, C., Kong, X., Xu, J., & Liu, Y. (2020). Learning to compensate for the drift and error of gyroscope in vehicle localization. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 852–857. https://doi.org/10.1109/IV47402.2020.9304715. https://ieeexplore.ieee.org/document/9304715/.

Zhou, B., Gu, Z., Gu, F., Wu, P., Yang, C., Liu, X., Li, L., Li, Y., & Li, Q. (2022a). DeepVIP: Deep Learning-Based Vehicle Indoor Positioning Using Smartphones. *IEEE Transactions on Vehicular Technology, 71*(12), 13299–13309. https://doi.org/10.1109/TVT.2022.3199507

Zhou, B., Wu, P., Gu, Z., Wu, Z., & Yang, C. (2022b). XDRNet: Deep Learning-based Pedestrian and Vehicle Dead Reckoning Using Smartphones. In *2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8. https://doi.org/10.1109/IPIN54987.2022.9918132. https://ieeexplore.ieee.org/document/9918132/.

Zhou, H., Zhao, Y., Xiong, X., Lou, Y., & Kamal, S. (2022c). IMU Dead-Reckoning Localization with RNN-IEKF Algorithm. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11382–11387. https://doi.org/10.1109/IROS47612.2022.9982087. https://ieeexplore.ieee.org/document/9982087/.