Vehicle Inertial Tracking via Mobile Crowdsensing: Experience and Enhancement

Yao Tong, Shuli Zhu, Xiaotong Ren, Qinkun Zhong, Dan Tao, Chi Li, Lei Liu, and Ruipeng Gao

***Abstract*— Nowadays, GPS and other global navigation satellite systems (GNSSs) have been widely developed, enabling accurate and convenient outdoor location-based services for vehicles. However, there is still a range of areas in an urban city that cannot be covered by satellites, e.g., large-scale tunnels, underground parking lots, and multilevel flyovers. Current vehicle inertial tracking methods always rely on dead-reckoning, but their performances are seriously affected by intrinsic fluctuations and external disturbances. Based on our experiments with thousands of smartphones, we summarize three threats for vehicle inertial tracking, i.e., arbitrary and unknown placements of smartphones during driving, volatile and inconstant noises on low-quality inertial sensors in commodity smartphones, and a diversity of smartphones and vehicles via crowdsensing. In this article, we explore a novel smartphone-based inertial learning framework to infer a vehicle's location in real time. It contains a coordinate transformation algorithm to yield vehicle's movements from smartphone's inertial readings, an inertial sequence learning model to train and infer the trajectory instead of double integrations, and a customized model refinement mechanism to improve the tracking accuracy for individual drivers. Extensive experiments on the DiDi ride-hailing platform in two large cities have proved the effectiveness of our solution. In addition, our approach has been deployed on DiDi real-world mobile applications in 7.59 million devices and conducted 4.26 billion location inferences every day.**

***Index Terms*— Coordinate transformation, customized learning, mobile crowdsensing, smartphone localization, vehicle inertial tracking.**

I. Introduction

OWING to the maturity and popularization of GPS devices, vehicular positioning has become ubiquitous for drivers, e.g., route planning, real-time navigation, and automatic driving. Nowadays, even rookie drivers can travel in unfamiliar areas with outdoor location-based applications. However, due to the rapid development of urban cities, drivers frequently enter enclosed environments where barely no GPS signals can penetrate, e.g., large-scale tunnels, multilevel overpasses, and underground parking garages. The number of direct line-of-sight (DLOS) satellites is extremely reduced, thus causing the delay of signal phase, reduction of load-noise ratio, multipath propagation, and other serious interference [1]. Drivers may get confused in such nondistinctive areas and could not find their way out. Fig. 1(a) shows a tunnel in Xiamen city where vehicles must shift lanes and make turns inside the tunnel, and Fig. 1(b) illustrates an underground parking garage in Beijing where drivers frequently forget where they park the car in such a maze-like structure.

Manuscript received November 13, 2021; revised February 4, 2022; accepted February 22, 2022. Date of publication March 4, 2022; date of current version March 24, 2022. This work was supported in part by Beijing NSF under Grant L192004, in part by NSFC under Grant 62072029 and Grant 61872027, in part by the DiDi Research Collaboration Plan, in part by the OPPO Research Fund, and in part by the Open Research Fund of the State Key Laboratory of Integrated Services Networks under Grant ISN21-16. The Associate Editor coordinating the review process was Yan Zhuang. *(Corresponding author: Ruipeng Gao.)*

Yao Tong, Shuli Zhu, Xiaotong Ren, Qinkun Zhong, and Ruipeng Gao are with the School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: tongyao@bjtu.edu.cn; zhushuli@bjtu.edu.cn; xiaotren@bjtu.edu.cn; qkzhong@bjtu.edu.cn; rpgao@bjtu.edu.cn).

Dan Tao is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: dtao@bjtu.edu.cn).

Chi Li and Lei Liu are with the Maps and Public Transportation Department, DiDi Corporation, Beijing 100089, China (e-mail: lichi@didiglobal.com; liuleifrey@didiglobal.com).

Digital Object Identifier 10.1109/TIM.2022.3156993

Fig. 1. Examples of vehicle tracking in GPS-blocked environments. (a) Tunnel in Xiamen. (b) Parking garage in Beijing.

Positioning without GPS signals is not a new topic, especially for pedestrians. With decades of research, mainstream indoor localization technologies always rely on Wi-Fi [2], [3] and other RF signatures [4]–[8], but there are a series of limitations when they are deployed for vehicles [1]. First, there exists serious instability and susceptibility of RF signal fingerprints by indoor disturbances [9], and it is time-consuming and labor-intensive to collect and calibrate such fingerprints at a large scale. Second, the RF signals are transmitted from certain IT infrastructures, which can be sparse or even nonexistent in enclosed environments. Such a high-cost, low-yield, and unstable positioning approach is not effective for ubiquitous vehicular location-based services.

To track vehicles in such GPS-blocked environments, modern ride-hailing platforms (e.g., Uber, Lyft, and DiDi) are focusing on deploying inertial dead-reckoning methods (e.g., the extended Kalman filter (EKF) [10]) by using smartphone's inertial sensors. However, drivers may place their smartphones with arbitrary placements inside vehicles, and there are obvious and inconsistent noises of inertial sensors in commodity smartphones, thus causing extreme location errors and poor user experience.

Based on the large-scale and crowdsourced traffic data from DiDi ride-hailing platform, we have conducted a series of experiments and investigated the limitations of tracking vehicles with inertial measurements from smartphones. Since the vehicle's movement is typically a combination of rotations and translations, we focus on two motion dynamics measured by smartphones, i.e., angular rates by the gyroscope and linear accelerations by the accelerometer. The results from thousands of smartphones have demonstrated three technical challenges: 1) drivers may place smartphones with arbitrary postures in a car; thus, the phone's coordinate system is not consistent with the vehicle; 2) inertial sensors are susceptible to intrinsic fluctuations and external disturbances with volatile and inconstant noises beyond regular distributions; and 3) the tracking accuracy differs distinctively on different smartphones and vehicles, even with the same type.

In this article, we aim to enable an accurate, robust, and real-time vehicle tracking solution. In addition, the lack of the Internet connectivity in GPS-blocked environments also requires it as a smartphone-only approach, i.e., all sensing and computation tasks are completed locally without the cloud. Instead of adopting traditional inertial dead-reckoning methods for vehicles, we propose a novel inertial learning framework by temporal convolutional networks (TCNs) and customize such a model for refinement training and location inference on individual smartphones.

This article is an extension of our proceedings paper published in the 27th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2021) [11]. Specifically, our contributions consist of the following.

1) *Experience:* We have conducted extensive experiments with the traffic data from a famous ride-hailing platform and analyzed current limitations and technical challenges, including the arbitrary and unknown placement of smartphones during driving, the volatile and inconstant inertial noises, and the diversity of crowdsourced smartphones and vehicles.
2) *Improvements:* We explore a coordinate transformation algorithm to yield a vehicle's motion dynamics via smartphone's inertial readings. We also propose an inertial sequence learning framework to train and infer vehicle's locations instead of double integrations. In addition, we customize the model retraining mechanism and derive more accurate trajectories for individual smartphones.
3) *Evaluations:* We collect a large-scale crowdsourced dataset in two large cities in China. Our results outperform traditional EKF-based tracking methods and other sequential learning models. In addition, our approach has been applied in the current DiDi ride-hailing platform with smartphones and conducted location inferences per day.

II. Background

Location-based mobile applications enable convenient vehicle tracking and navigation services at the city level, by GPS, or other global navigation satellite system (GNSS) signals inside smartphones. Nevertheless, with the rapid development of urban cities, drivers frequently enter GPS-blocked environments, such as large-scale tunnels, multilevel overpasses, underground parking lots, and urban canyons. The lack of satellite perception seriously affects the driving experience, e.g., getting confused and disoriented by the location gap between the practical position at the tunnel exit and the inertial inferenced position (see Fig. 2).

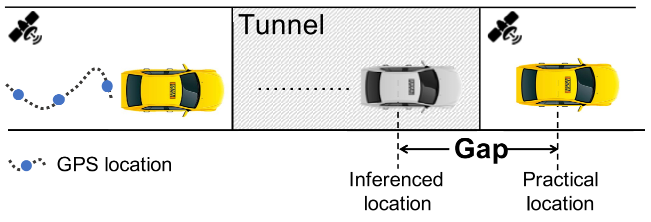


Fig. 2. Location gap between practical position at tunnel exit (by GPS) and the inferenced position (by inertial tracking).

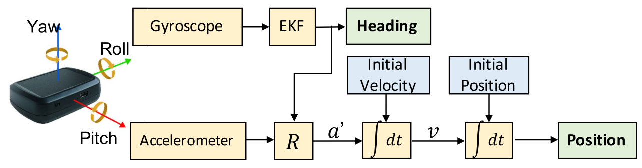


Fig. 3. Architecture of traditional inertial dead-reckoning. R is the acceleration projection matrix, and a' denotes the 2-D accelerations on the ground.

To provide location awareness in such environments, service providers mainly deploy on strap-down inertial navigation systems (INSs) for vehicle inertial dead-reckoning, via micro-machined electromechanical systems (MEMSs) inside smartphones. They exploit the smartphone's inertial data as inputs, i.e., the three-axis linear accelerations by accelerometer and the three-axis angular rates by the gyroscope, both measured at the phone's coordinate system. Instead of directly double integrating accelerations into distance ($\Delta x_t = \int\int a_t dt dt$), they estimate the vehicle's heading H_t, project the phone's accelerations as its forwarding accelerations $a'_t = H_t \cdot a_t$, and then compute the location variations as $\Delta x_t = \int\int a'_t dt dt$ for vehicle tracking.

In practice, drivers may place the smartphones with arbitrary postures in the vehicle; thus, the phone's coordinate system (X^p, Y^p, Z^p) is not always aligned with the vehicle's (X^v, Y^v, Z^v). The EKF is a widely used solution to estimate a smartphone's posture (see Fig. 3). Specifically, the EKF algorithm initializes the vehicle's attitude and motion models, and leverages the gyroscope to continually update the vehicle's corresponding states due to practical measurements, thus deriving the heading direction H_t.

However, due to the low quality of inertial sensors and the diversity of crowdsourced smartphones, extreme noises and outliers appear frequently, and they are easily accumulated to unbounded location errors.

III. Observation and Challenges

In this section, we conduct a series of experimental studies on large-scale crowdsourced datasets, investigate the impacts

TABLE I
DATASET INFORMATION

Source	Crowdsouced by DiDi	Dedicated by ourselves
Location	Beijing and Shenzhen	Beijing
Time	December 1 ~ 30, 2020	April 1 ~ May 30, 2021
Distance	(17195 + 5591) km	62 km
Smartphones	(876 + 829) phones	6 phones
Ground truth	GNSS (1Hz)	Dedicated INS (100Hz)

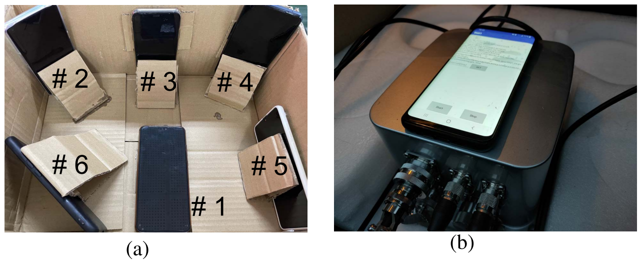


Fig. 4. Dedicated measurement by a mold with six smartphones. (a) Mold with six phones. (b) Dedicated IMU device.

of smartphone inertial sensors on tracking vehicles, and present our observations.

A. Data Source

In order to explore the influence of inertial sensors for crowdsourced vehicle tracking, we have collected millions of driving trajectories by the DiDi ride-hailing platform. Specifically, we classify our data source into two categories, i.e., the crowdsourced dataset and the dedicated dataset. Details are shown in Table I.

1) Crowdsourced Dataset: Our large-scale real-world crowdsourced dataset is collected by the DiDi platform in two urban cities (Beijing and Shenzhen) in China between December 1 and 30, 2020. The training data are gathered from 876 phones during 308 driving hours, covering 17 195-km distances. The test data are from 829 phones during 191 driving hours, covering 5591-km distances.

2) Dedicated Dataset: Since the crowdsourced data lack the ground truth of the smartphone's posture, we build a mold to hold six smartphones with different placements, as shown in Fig. 4(a). Meanwhile, we leverage a dedicated IMU device [see Fig. 4(b)] in the same vehicle and collect its measurements as the inertial ground truth.

B. Observations

The vehicular motion is typically a combination of rotation and translation movements. Thus, we investigate two motion factors, i.e., angular velocities measured by the gyroscope and linear accelerations measured by the accelerator, both in the smartphone's coordinate system. In order to explore the errors caused by inertial noises, we fix a smartphone and align it with the vehicle; thus, the smartphone's inertial readings are approximated as the vehicles.

1) Gyroscope: The gyroscope measures real-time angular rates (w_X^p, w_Y^p, w_Z^p) around the X^p-axis, the Y^p-axis, and the Z^p-axis in the smartphone coordinate system, respectively.

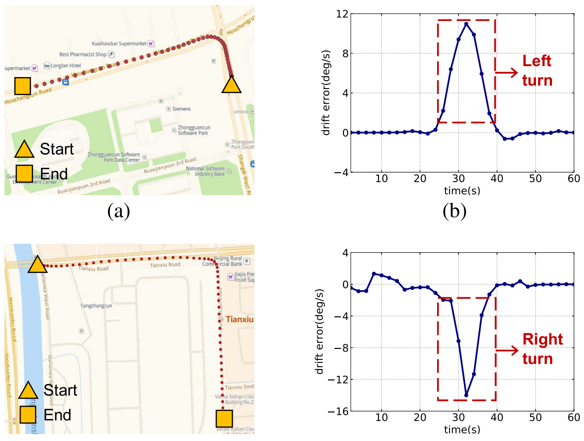


Fig. 5. Two turns with GPS trajectory and the corresponding gyroscope drift error in 1 min. (a) Left turn. (b) Drift error in left turn. (c) Right turn. (d) Drift error in right turn.

TABLE II
SKEWNESS AND KURTOSIS IN DIFFERENT ROADS

Phones	Left turn		Right turn		Straight	
	skew	kurt	skew	kurt	skew	kurt
#1	1.94	3.96	-2.79	8.04	1.01	7.21
#2	1.96	4.06	-2.81	8.17	1.01	7.23
#3	1.94	3.96	-2.79	8.03	1.01	7.2
#4	1.93	3.89	-2.80	8.06	0.99	7.29
#5	1.96	4.05	-2.81	8.15	1.0	7.09
#6	1.95	3.98	-2.79	8.04	1.01	7.2

Theoretically, with continuous integration on angular velocities, we derive the vehicle's deading direction for tracking. However, due to the low quality of MEMS's gyroscope, its readings are affected by numerous noises, such as the constant bias, thermomechanical white noise, and flicker noise [12].

To investigate the impact of gyroscope errors on rotational measurements, we perform continuous integration on angular rates in straight road segments and turning road segments, respectively. Straight segments are defined as their angular variation less than 10° within 1 min, and turning segments denote the ones with more than 45°. The ground truth is measured by a dedicated IMU device, as described in our dedicated dataset. We compared their angular drift errors (the angular difference between ground truth and integration results) in unit time.

An interesting finding is that the drift errors of left turn and right turn have the same amplitude but inverse value, as depicted in Fig. 5. Based on the skewness and the kurtosis value in Table II, we observe that no matter driving on a turning road or straight road, neither the skewness nor the kurtosis of drift errors is closed to zero. In addition, six phones have the same potential manifestation. All phenomena demonstrate that the gyroscope's drift errors do not follow the normal distribution.

2) Accelerometer: Accelerometer measures the three-axis linear accelerations (a_X^p, a_Y^p, a_Z^p) in smarpthone's coordinate system. The types of accelerator' drift errors [12] are analogous to the gyroscope, expect the arising errors due to the double integrations for distance.

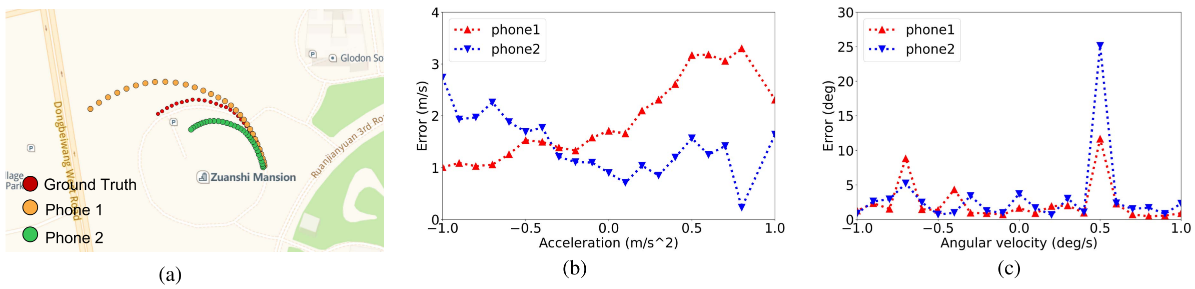


Fig. 6. Velocity and orientation comparison between two smartphones with the same type of MEMS sensors. (a) Trajectories on the map. (b) Linear acceleration errors. (c) Angular rate errors.

TABLE III
SKEWNESS AND KURTOSIS IN ACCELERATING AND DECELERATING

Velocity	Skewness		Kurtosis	
	skew	z-score	kurt	z-score
Accelerating	0.24	1.49	-0.74	-2.28
Decelerating	0.19	2.0	-0.68	-3.51

To investigate the impact of linear accelerations, we extract the accelerating and decelerating segments within 5-s intervals in straight roads and calculate the integrated drift errors on six smartphones. The ground truth is also supported by the dedicated IMU device. Although its skewness and kurtosis are closed to zero (see Table III), their corresponding z-sore does not satisfy the hypothetical conditions ($-1.96 \leq$ z-score ≤ 1.96), while the inspection level $\alpha = 0.05$. As a result, the drift errors of the accelerometer also disobey the normal distribution.

3) Inertial Sensor With the Same Type: In order to analyze the impact of the type of inertial sensors in different smartphones, we deploy two smartphones with the same type in the same vehicle. We leverage a 10-s sliding window to process the temporal sequence and calculate its accumulated errors. As depicted in Fig. 6, the linear acceleration errors on Phone #1 and Phone #2 have an opposite error trend. When acceleration changes in $(0, 1)$, Phone 1 gets a lower error, while Phone 2 obtains much more. Their maximum orientation errors are also different when the average angular velocity is at 0.3 °/s. The experimental results demonstrate that different smartphones with the same type perform distinctively in accelerations and angular rates, even at the same time in the same vehicle.

C. Challenges

Based on our observations, the technical challenges of city-level vehicle inertial tracking are summarized as follows.

1) Arbitrary Posture of Smartphones: Drivers have their own preference to place the smartphone in a vehicle; thus, the posture of the smartphone in the car is arbitrary, and its coordinate system cannot always align with the vehicles. The exiting EKF-based heading estimation method is affected by the low-quality MEMS sensors in smartphones.

2) Inconstant Inertial Errors: The inertial errors are varying dramatically via crowdsensing. The drift errors of the accelerometer or gyroscope do not follow the normal distribution, and the direction of angular errors is the inverse between left and right turns. Thus, we cannot adopt a general error distribution model to fit all smartphones. A possible approach is to leverage the deep learning method for temporal sequence learning, which captures long-term dependencies instead of double integrations.

3) The Diversity of Smartphones: Due to the different manufacturers of commodity inertial sensors, we observe that different smartphones proverbially achieve various localization accuracies. A straightforward solution is to customize the parameters for each type of smartphone. However, to our surprise, two smartphones with the same type of MEMS sensors still have distinct localization performances even in the same car.

D. Design Guidelines

To address the above challenges, we propose a novel vehicle inertial learning framework. As depicted in Fig. 7, it is comprised of three phases.

1) *Coordinate Transformation:* We explore a PCA-based pose estimation algorithm to derive smartphone's placement in the car and further transform the smartphone's inertial dynamics into the vehicle's (see Section IV).
2) *Inertial Sequence Learning:* We propose a deep sequence learning model to settle the inconstant noises of inertial measurements and produce vehicle's movements (see Section V).
3) *Customized Model Refinement:* After training a general model among crowdsourced smartphones, we refine the customized model of individual smartphones and derive accurate trajectories (see Section VI).

IV. COORDINATE TRANSFORMATION

Since drivers may place their smartphones with arbitrary postures during driving, the phone's coordinate system is not always consistent with corresponding vehicles; thus, its three-axis accelerations (a_X^p, a_Y^p, a_Z^p) and angular rates (w_X^p, w_Y^p, w_Z^p) cannot be directly used as vehicle's motion dynamics. Suggesting every driver placing the phone in an absolute vertical posture is unconscionable and offensive. In this section, we explore a principal component

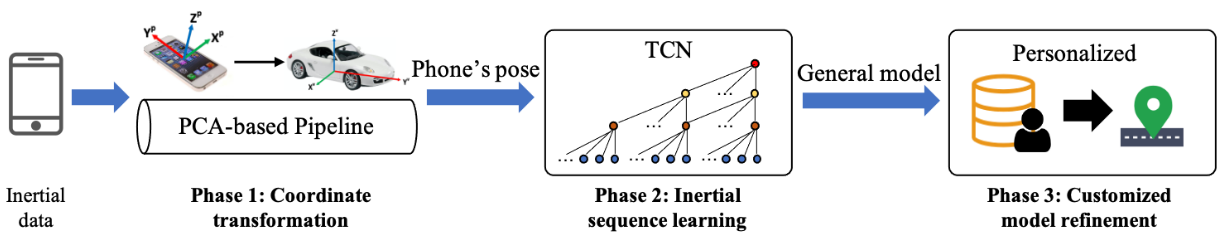

Inertial data
PCA-based Pipeline
Phone's pose
TCN
General model
Personalized
Phase 1: Coordinate transformation
Phase 2: Inertial sequence learning
Phase 3: Customized model refinement

Fig. 7. System overview with three phases, i.e., coordinate transformation, inertial sequence learning, and customized model refinement.

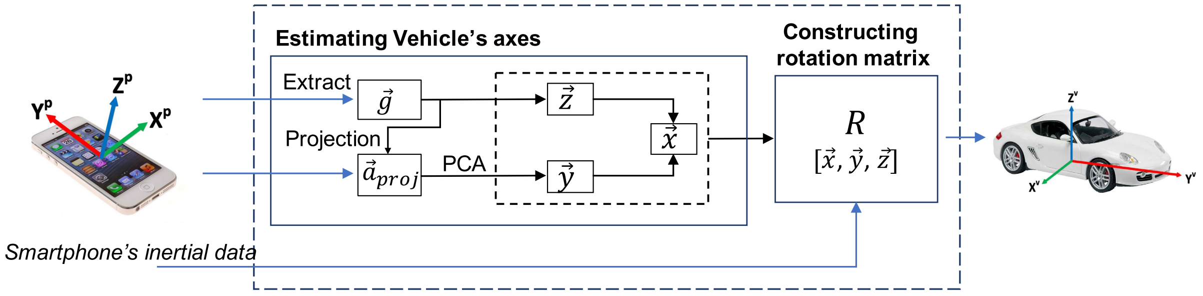

Estimating Vehicle's axes
Constructing rotation matrix
Extract
Projection
PCA
$\vec{g}$
$\vec{a}_{proj}$
$\vec{z}$
$\vec{x}$
$\vec{y}$
R
$[\vec{x}, \vec{y}, \vec{z}]$
Smartphone's inertial data

Fig. 8. Workflow of phone pose estimation, consisting of two steps: estimating vehicle's three axes in corresponding smartphone's coordinate system and constructing the relative rotation matrix to transform the inertial dynamics.

analysis (PCA)-based motion transformation algorithm.[1] As shown in Fig. 8, it is comprised of two phases: estimating the vehicle's three axes and constructing the corresponding rotation matrix.

A. Vehicle's Axes Estimation

We propose a three-step algorithm to estimate the vehicle's three axes in contrast to the smartphone's coordinate system.

Step 1 (Z^v-Axis): When vehicles are driving on the horizontal plane, the Z^v-axis is exactly opposite with the gravity direction. Since there are always slight vibrations during driving, we adopt a low-pass Butter-worth filter to denoise the signal and then normalize it to a unit vector z as the Z^v-axis.

Step 2 (Y^v-Axis): Intuitively, vehicle's accelerations in straight driving indicate its forwarding direction Y^v-axis. In practice, we observe that there are heavy noises due to inevitable jolts; thus, we remove the turning regions (whose angular variation is more than 45° in 1 min) and project smartphone's three-axis accelerations $\boldsymbol{a}^p$ onto the horizontal plane by the unit vector z (as shown in Fig. 9)

$$\boldsymbol{a}_{\text{proj}} = \boldsymbol{a}^p - (\boldsymbol{a}^p \cdot z)z \tag{1}$$

where $\boldsymbol{a}_{\text{proj}}$ is the projected acceleration on horizontal plane.

Next, we aim to derive a vector $\boldsymbol{w}$ on horizontal plane, which optimally depicts the variation of accelerations, i.e., $\boldsymbol{w}^\top \boldsymbol{a}_{\text{proj}}$ (the right part in Fig. 9). We regard $\boldsymbol{w}$ as vehicle's Y^v-axis. Based on the orthogonality of a coordinate system, the largest variance indicates the most separation. Therefore,

[1]This algorithm is adopted only when drivers initiate a ride or change their smartphone's placement during driving. Specifically, we observe that the variation of smartphone's placement causes sharp accelerations along all three axes; thus, a simple threshold on the energy can identify such event.

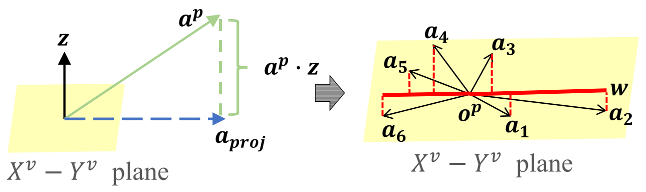


Fig. 9. Phone's acceleration projection and vehicle's forwarding estimation.

our optimization target is defined as

$$\begin{aligned}&\max_{\boldsymbol{w}}\ \operatorname{tr}\left(\boldsymbol{w}^\top \boldsymbol{a}_{\text{proj}} \boldsymbol{a}_{\text{proj}}^\top \boldsymbol{w}\right)\\ &\text{s.t. } \boldsymbol{w}^\top \boldsymbol{w} = I\end{aligned} \tag{2}$$

where tr(.) denotes the variance value.

We propose a PCA-based algorithm to derive the orthometric vector with the largest variance value, i.e., the max principal component. Its details are presented as follows.

Step 2.1 (Normalization): For the projected accelerations $a_i = (x_i, y_i, z_i)$, $i = 1, \ldots, k$, where k is the sequence length, we normalize them along each axis

$$\bar{a}_i = \left(x_i - \frac{1}{k}\sum_{i=1}^{k} x_i, y_i - \frac{1}{k}\sum_{i=1}^{k} y_i, z_i - \frac{1}{k}\sum_{i=1}^{k} z_i\right). \tag{3}$$

Meanwhile, its covariance matrix is constructed as

$$\text{Cov} = \frac{1}{k} A_{k*3} A_{k*3}^\top \tag{4}$$

where $A_{k*3} = [\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_k]$.

Step 2.2 (Matrix Decomposition): There are two common methods of matrix decomposition: 1) eigenvalue decomposition and 2) singular value decomposition (SVD). However, the first requires that the input should be square matrix, which is not consistent with our source matrix $A_{3*k} (k \gg 3)$. Therefore, we leverage the SVD method to decompose the acceleration matrix, i.e., $A_{k*3} \approx \mathcal{U}_{k*r} \Sigma_{r*r} \mathcal{V}_{r*3}$, where $\mathcal{V}_{r*3}$ denotes the right singular vector, $\mathcal{U}_{r*3}$ is the left singular vector, and Σ_{r*r} is a rectangular diagonal matrix. In addition, the largest eigenvalue γ in Σ corresponds to the jth line vector of $\mathcal{V}$, whose transpose is the expected Y^v-axis $\boldsymbol{y}$.

Step 3 (X^v-Axis Estimation): Based on the Y^v-axis and the Z^v-axis, we derive the rest X^v-axis as their cross-product

$$\boldsymbol{x} = \boldsymbol{y} \times \boldsymbol{z}. \tag{5}$$

Thus, $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ are orthogonal with the others.

B. Rotation Matrix Construction

Since every axis vector denotes an orthogonal direction for dimensionality reduction, we concatenate three-axis vectors in columns to derive the rotation matrix $R_v^p = [\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]$, which presents the rotation relationship from smartphone's coordinate system to the vehicles. Next, we transform the smartphone's inertial readings into vehicle's motion dynamics in its own coordinate system, i.e.,

$$\boldsymbol{a}^v = \boldsymbol{a}^p R_v^p \tag{6}$$

$$\boldsymbol{w}^v = \boldsymbol{w}^p R_v^p. \tag{7}$$

In addition, in order to evaluate the accuracy of the estimated poses, we need to calculate the Euler angles of smartphones in contrast to vehicles. Note that the rotation matrix can be decomposed of different Euler angles with different axis rotation orders, e.g., $x - y - z$, $z - y - x$, or $y - x - z$. Besides, the positive direction of rotation is defined based on the right-hand rule. Specifically, suppose that we rotate the smartphone's coordinate system (X^p, Y^p, Z^p) to align with the vehicle's (X^v, Y^v, Z^v) in the $z - y - x$ order; the rotation matrix R_v^p is decomposed as

$$\begin{aligned} R_v^p &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = R_z(\gamma) * R_y(\beta) * R_x(\alpha) \\ &= \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \\ &\quad \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \end{aligned} \tag{8}$$

where R_x, R_y, and R_z denote the rotation along the X^p-, Y^p-, and Z^p-axes, respectively. Thus, the Euler angles (α, β, γ) are derived $\gamma = \arctan 2(-r_{12}, r_{11})$, $\beta = \arcsin(r_{13})$ and $\alpha = \arctan 2(-r_{23}, r_{33})$ with the $z - y - x$ rotation order.

V. Inertial Sequence Learning

After coordinate transformation, we derive the inertial dynamics in the vehicle's coordinate system. Since the inertial error is always volatile and inconstant, a general error distribution model is hard to construct for crowdsourced smartphones. Instead of conducting inertial dead-reckoning, we explore an inertial sequence learning framework to produce a neural network that minimizes the differences between actual inertial readings and corresponding trajectories.

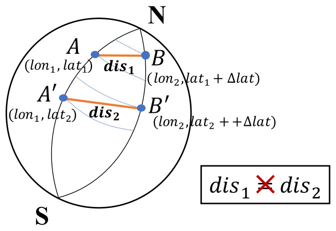


Fig. 10. Variations of longitude and latitude are the same for two traces AB and $A'B'$, while their spherical distance dis_1 and dis_2 are different.

A. Model Formulation

Intuitively, we aim to estimate the location variance $(\Delta \text{lon}_{0:t}, \Delta \text{lat}_{0:t})$ during each time interval. However, the spherical distance on the same longitude varies in different latitudes (see Fig. 10). Instead, we aim to predict the speed variations and split them into the velocity and bearing $(\Delta v_{0:t}, \Delta o_{0:t})$. Specifically, $\Delta o_{0:t}$ is an intricate nut to crack; thus, we suppose that the angular variation of bearing is within $(0, 360)$, and the clockwise direction is positive, and vice versa.

The original inputs from smartphones consist of the three-axis accelerations (a_X^p, a_Y^p, a_Z^p), three-axis angular velocities (w_X^p, w_Y^p, w_Z^p), and three-axis gravity accelerations (g_X^p, g_Y^p, g_Z^p); all are collected at 50 Hz. When vehicles are driving in outdoor scenarios, we leverage the GPS location (longitude and latitude), speed, and bearing (v, o) as the ground truth (1 Hz) to train the model.

In order to deploy the inference model in smartphones, we cannot directly concatenate the raw 9-D inertial data as-is, which involves complex neural networks to capture such an amount of data. Instead, we extract the most efficient features instead of raw inertial data as model inputs. Specifically, the features we exploit include: 1) the initial speed init_speed and bearing init_bearing (vehicle's heading direction) at the initial point, i.e., the last valid GPS information and 2) the accelerometer features, gyroscope features, and gravity features, and we calculate the corresponding mean, min, max, std, var, and sum values at a 1-s interval. Therefore, the data amount of a 1-s inertial sequence is reduced from $3 \times 3 \times 50 = 450$ to $3 \times 3 \times 6 = 54$, thus decreasing the memory to about only 1/8 of the original.

B. TCN Architecture

We use a TCN framework with residual blocks to learn the inertial sequence. Although recurrent neural networks (RNNs), e.g., the long short-term memory (LSTM) and the gated recurrent unit (GRU), have become the most popular architectures for sequence modeling, they are inefficient on mobile devices due to a large number of parameters. Instead, we adopt the TCN, which is more suitable for deployment on smartphones with a long-range receptive field. Specifically, the residual blocks used in TCN have been proved to be an effective way to train deep networks, which enables networks to transmit information in a cross-layer manner.

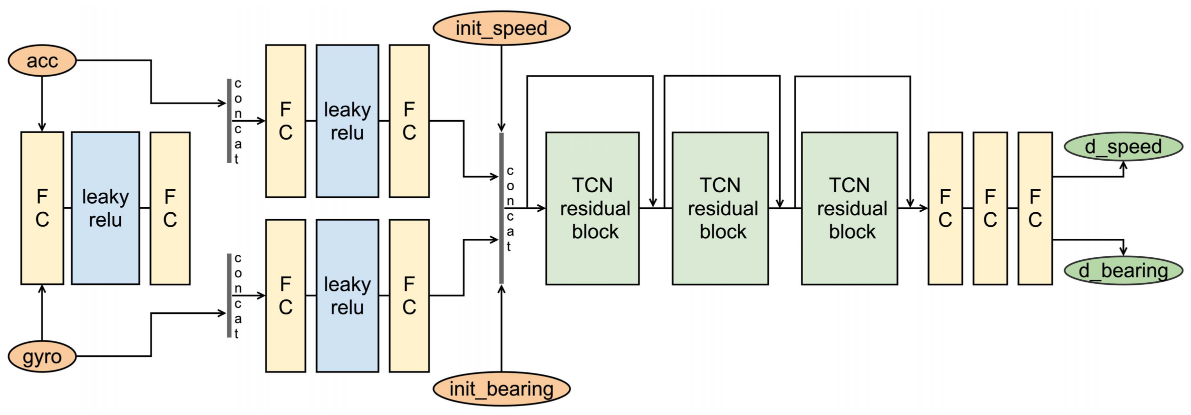


Fig. 11. Our TCN framework for vehicle inertial tracking, including nine layers of FC layers and three TCN residual blocks.

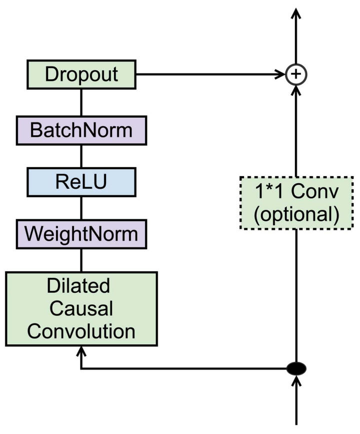


Fig. 12. Implementation details of a TCN residual block with dilated causal convolution, weight normalization, ReLU, batch normalization, spatial dropout, and a residual connection.

As shown in Fig. 11, the hyperparameters of our model include nine layers of a fully connected (FC) network and three TCN residual blocks (batch size of 128 and the learning rate of 0.001). We use the rectified linear unit (ReLU) activation function in each TCN residual block, whereas the Leaky-ReLU is used for the first six layers of the FC network. In each TCN residual block (see Fig. 12), we apply the dropout rate of 0.5 to mitigate overfitting. Meanwhile, we use the weight normalization and batch normalization to accelerate model convergence and the Adam optimizer to iteratively update network weights.

Since our objective is to estimate both velocity and orientation estimates, a straightforward method is to use the weighted *SmoothL1Loss* as loss metric, which is less sensitive to outliers than the mean squared error (MSE) and prevents exploding gradients in extreme cases. The *SmoothL1Loss* is a popular metric to for loss computation, i.e.,

$$L(\hat{x}, x) = \begin{cases} |\hat{x} - x| - 0.5, & |\hat{x} - x| > 1 \\ 0.5|\hat{x} - x|^2, & |\hat{x} - x| \leq 1. \end{cases} \tag{9}$$

However, when combining velocity and bearing estimates, it leads to a series of problems. First, the variation of orientation in unit time is much larger than velocity, as shown in Table IV. Thus, the model inclines to calibrate orientation mistakes rather than the velocity. Besides, the loss value seriously accumulates as time passes by, as shown in Fig. 13.

TABLE IV
COMPARISON OF THE DISTRIBUTION OF THE SPEED AND THE BEARING VARIATIONS DURING TRAINING

Percentile	Mean	50th	80th	90th
Δv(m/s)	0.50	0.35	0.81	1.14
Δo(deg)	2.55	0.70	3.10	6.80

Therefore, we customize the loss computation. Specifically, we add an enhancement factor α on velocity item to balance the two subnetworks and devise a time attenuation function $w(t)$ to reduce loss accumulation, i.e.,

$$w(t) = \exp\left(\frac{-t^2}{2\sigma^2}\right) \tag{10}$$

where σ is time attenuation factor and t is the timestamp. The intuition of the weight $w(t)$ is to concentrate on features from current time series. Finally, our customized loss function is presented as

$$\text{Loss} = \sum_{t=1}^{N} w(t)[\alpha L(\Delta\hat{v}_t, \Delta v_t) + L(\Delta\hat{o}_t, \Delta o_t)]. \tag{11}$$

VI. CUSTOMIZED MODEL REFINEMENT

With the large-scale database collected via crowdsensing, we have established a general location inference model among all users. However, based on our observation in Section III, vehicle's tracking accuracy differs significantly even on the same type of smartphones (see Fig. 6). A straightforward approach is to customize the location inference model to fit individual smartphones. Though appealing, it has two inherent limitations. First, it will lead to privacy concerns if people upload their personal driving trajectories via the Internet. In addition, the deep neural networks in customized learning may cost dozens of minutes for training on individual smartphones. Thus, the realization of customized learning becomes a nontrivial journey for both on-cloud training and on-device training.

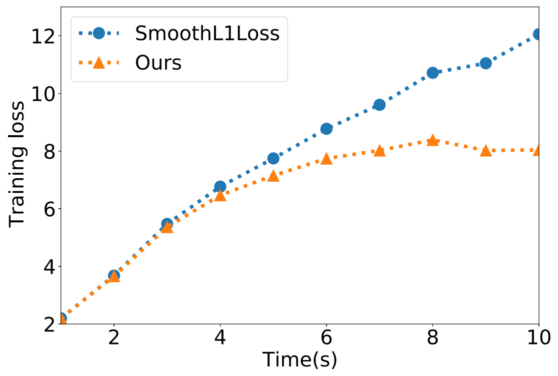


Fig. 13. Training comparisons between SmoothL1Loss and our customized loss metric. The data window is set as 10 s.

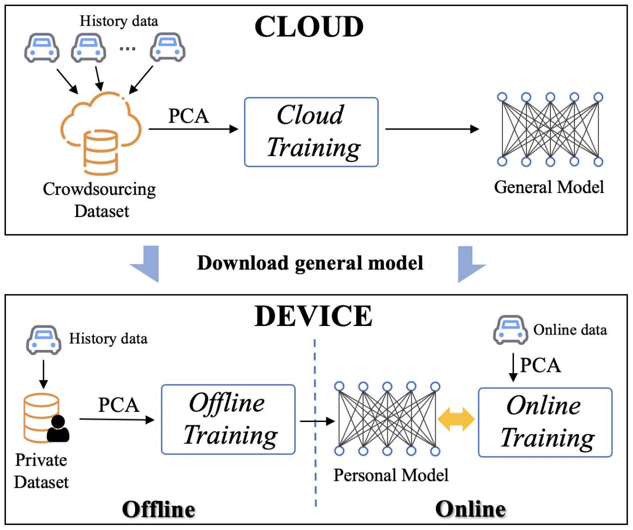


Fig. 14. Mechanism of customized model refinement. We train a general model via mobile crowdsensing and refine it via personal online data.

To address the above issues, we propose a customized model refinement mechanism to address the diversity of crowdsourced smartphones. As depicted in Fig. 14, its training process comprises three phases:

1) *Cloud Training:* We use the crowdsourced dataset to train a general location tracking model. This phase is typically performed on the cloud. The trained model is called a general model. After achieving the expected accuracy on the crowdsourced dataset, we dispatch this general model to all users' devices.
2) *Off-Line Training:* On each mobile device, we leverage its historical personal data and retrain the general model. The customization is mainly achieved by synthesizing the dataset that users refuse to upload to the Internet. Finally, we produce and deploy the customized model on individual smartphones.
3) *Online Refinement:* During the vehicle's outdoor driving, each smartphone collects the latest inertial data with reliable GPS locations. We use such data to refine the user-specific model in real time and continuously update our personal inference model.

Based on this customized refinement mechanism, we derive a more accurate tracking model for individual drivers without incurring privacy concerns. Note that off-line training and online training are both scheduled when devices are in charging mode, minimizing the impact of battery usage.

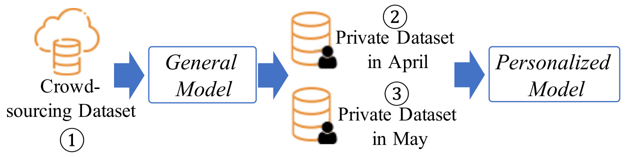


Fig. 15. Multistage model training mechanism. We utilize ① (the crowdsourced dataset) to produce the general model, refine it with ② (the private dataset in April) for off-line training, and further refine it with ③ (the private dataset in May) for online training.

TABLE V
METHODOLOGY FOR MODEL TRAINING AND TESTING

Models	Training set	Testing set
General model	①	②
Offline model without PCA	① → ②	②
Offline model with PCA	① → ②	②
Offline model with PCA	① → ②	③
Online model with PCA	① → ② → ③	③

VII. Evaluation

A. Methodology

1) *Training and Testing Datasets:* As we illustrated in Section III, our crowdsourced data are gathered via the DiDi ride-hailing platform in Beijing and Shenzhen, China. The detailed description is depicted in Table I. As a supplement, our training dataset is at the duration of 10 s, while the testing dataset is at 60 s. To distinguish the crowdsourced dataset and the dedicated dataset, we mark the crowdsourced dataset as ①, the dedicated dataset in April as ② for off-line training with the historical data, and the dedicated dataset in May as ③ to simulate online training and inference with the latest data (as shown in Fig. 15). In addition, the datasets ② and ③ are both collected by six smartphones with different placements in the mold [as shown in Fig. 4(a)]. The ratio for training/testing split in each dataset is 3:1. Detailed descriptions for customized model training are shown in Table V.

2) *Compared Alternatives:* We compare our performance with EKF, LSTM, and GRU algorithms for vehicle inertial tracking. The EKF approach is currently applied by many ride-hailing platforms to track vehicles without GPS, and we adopt a maximum likelihood estimation (MLE) solution to produce its parameters under the normal distribution assumption. The LSTM and GRU are implemented by replacing the TCN block with LSTM (*torch.nn.LSTM*) and GRU (*torch.nn.GRU*). Their special network parameters include *hidden_size* = *256* and *hidden_layer* = *3.*

3) *Training Details:* During the training process, we leverage the early stopping method that can prevent overfitting. When the loss on the testing dataset remains continuously stable after several times, we stop training the model. Our training and validation losses are stabilized at 1.55 and 1.67 after 50 epochs.

B. Phone Pose Estimation

We have derived the rotation matrix between the vehicle's coordinate system and the phone's, and constructed the

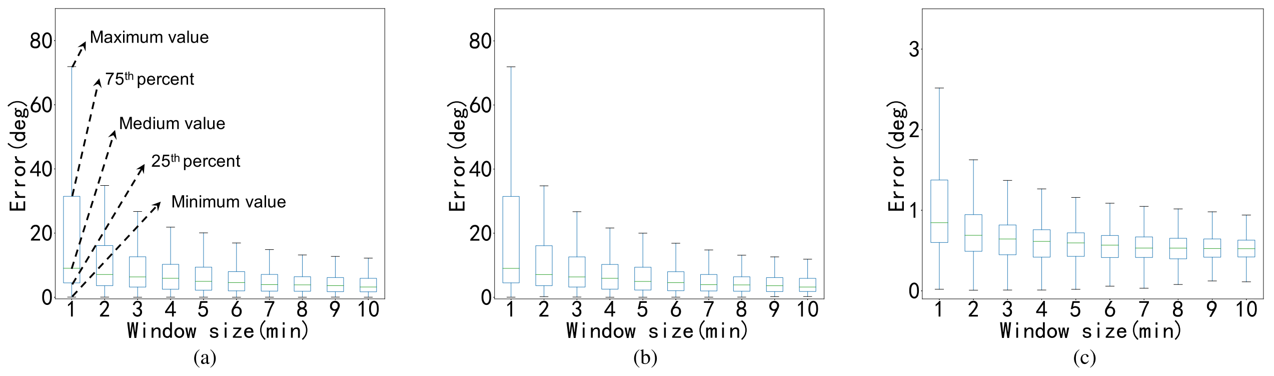


Fig. 16. Pose estimation errors along three axes with different window sizes. We use the box diagram to depict the errors at the maximum, 75th percentile, medium, 25th percentile, and minimum values, respectively. (a) X-axis. (b) Y-axis. (c) Z-axis.

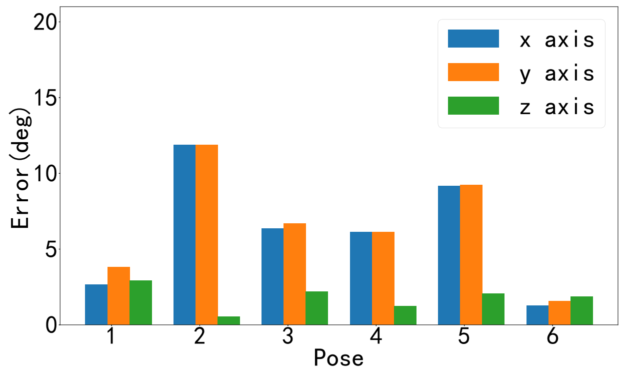


Fig. 17. Pose estimation errors for six smartphones with different placements. The ground-truth placements of smartphones are shown in Fig. 4(a).

corresponding Euler angles via the $z - y - x$ rotation order. Here, we measure the pose estimation accuracy with the six different placements in the mold [as shown in Fig. 4(a)]. These poses hopefully cover all common driving scenarios.

1) PCA Time Window: An effective and short-range time window is crucial for the accuracy and real-time posture prediction. In order to evaluate the appropriate time window for pose estimation, we extract 20 trajectories on straight roads and test the window size from 1 $\sim$ 10 min with a sliding window of one second. We leverage the box diagram to display pose estimation errors at the maximum, 75th percentile, medium, 25th percentile, and minimum values, respectively. As shown in Fig. 16(a)–(c), the pose estimation errors in the X-axis and the Y-axis reduce significantly with the increasing window size, while the errors in the Z-axis remains highly accurate (less than 3°). However, a long time window impedes real-time performance in case drivers move their smartphones. To balance the accuracy and delay of pose estimation, we adopt a 5-min time window in our system.

2) Posture Accuracy: We use the 5-min time window to predict six different placements with our PCA-based pose estimation algorithm. Fig. 17 shows the accuracy along three axes. We observe that the estimated pose errors are consistently smaller than 12° for all six poses. These results indicate that our pose estimation algorithm is effective and robust to transform the inertial dynamics from smartphones to vehicles.

TABLE VI
MAE ON LOCATION INFERENCE WITH DIFFERENT LOSS METRICS (M)

Loss metric	Beijing		Shenzhen	
	30 seconds	60 seconds	30 seconds	60 seconds
SmoothL1Loss	76.76	161.71	75.06	158.14
Ours	**23.64**	**71.19**	**21.17**	**66.23**

C. Inertial Sequence Learning

1) Loss Optimization: Directly regarding the *SmoothL1Loss* as the loss metric leads to serious error accumulations and longer time for convergence (as shown in Fig. 13). In addition, it cannot balance the weights of speed and bearing estimates. Instead, our customized loss metric involves the time attenuation function and enhancement factor to speed up the training process. Besides, we evaluate the continued tracking errors in 30- and 60-s durations in Table VI in both Beijing and Shenzhen, respectively. Compared with the *SmoothL1Loss*, our customized loss metric significantly reduces the tracking errors in location inference (almost by two-thirds).

2) Tracking Accuracy: We aim to evaluate the tracking accuracy of vehicles driving after a certain duration, e.g., 30 s in Fig. 18 and 60 s in Fig. 19, respectively. The mean absolute error (MAE) is used as the loss metric, which compares the difference between the ground truth and our predicted location after such duration. Since there are few opportunities to calibrate a vehicle's location in GPS-blocked environments, longer tracking duration inevitably accumulates tracking errors with higher MAE values. Compared with EKF, GRU, and LSTM, we observe that our TCN-based approach achieves the least MAE values, both in the short and long terms. Besides, there are only 1/3 parameters of our inference model compared with GRU and LSTM; thus, ours is more suitable for deployment on smartphones. Fig. 20 demonstrates our effectiveness in an overpass in Beijing and a tunnel in Shenzhen for real-time tracking without GPS. Compared with the EKF that is currently used by the DiDi platform, our method performs more accurate trajectories.

D. Online Customized Learning

We first evaluate the accuracy of the customized model during off-line training and then leverage the online driving

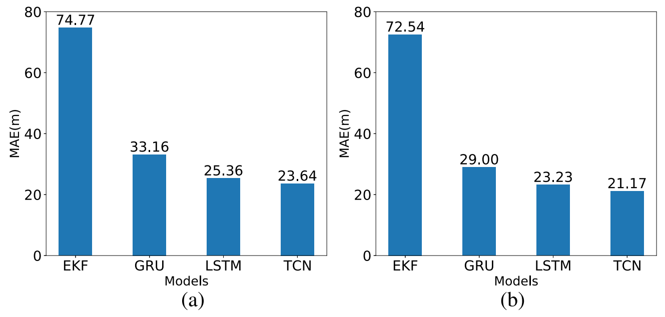


Fig. 18. Tracking vehicles after 30-s duration in (a) Beijing and (b) Shenzhen.

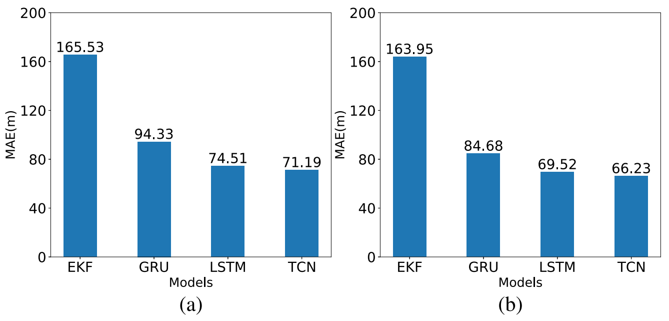


Fig. 19. Tracking vehicles after 60-s duration in (a) Beijing and (b) Shenzhen.

data to evaluate the accuracy improvement by online training.

1) Effectiveness of Pose Estimation and Off-Line Customized Learning: In the off-line training process, we comprehensively compare the model performance based on two guidelines: 1) transforming coordinate with PCA-based pose estimation and 2) retraining customized model with user-specific data in April. Finally, we achieve four alternatives, as shown in Table V.

Fig. 21 denotes the substantial improvement contributed by pose estimation and off-line customized learning. When we leverage the off-line user-specific data to retrain the general model and produce the customized model (off-line model without PCA), it reduces the localization errors from 13.114 to 11.437 m (↓12.79%) for 10-s inference. In addition, if we combine the pose estimation and customized retraining together (off-line model with PCA), it further reduces the tracking errors into 8.0 m (↓38.99%). The experimental results demonstrate that both pose estimation and off-line customized retraining improve tracking accuracy.

2) Effectiveness of Online Customized Learning: Off-line customized training, i.e., training and testing with the historical user-specific data in April, achieves respectable improvements. In addition, online training is able to update the customized model by leveraging the latest trajectory from a specific driver. As shown in Fig. 22, the online training (online model with PCA) further reduces the tracking errors from 7.802 m by the off-line model to 6.818 m (↓12.61%). It indicates that, although our model has been well customized by off-line training with historical user-specific data, it can still be enhanced by the online data. This inspires us to continuously exploit the latest inertial data into customized training and fine-tune our model to improve tracking accuracy.

E. Real-World User Experience

Due to our collaboration with the DiDi ride-hailing platform, it has applied our model to its real-world mobile application in 7.59 million devices (3.43 million daily active devices). Specifically, based on their statistics, our model has conducted 4.26 billion location inferences per day and costs 21.2 ms as average inference delay.

In addition, we also evaluate the practical user experience of our model in real-world tunnels. Specifically, we calculate the distance between the practical location at tunnel exit (by GPS) and our last inferenced location and regard a location gap if the distance is larger than 65 m and an extreme gap if the distance is larger than 130 m. Finally, we compute the gap rate (percentage of orders with gaps) and the extreme gap rate (percentage of orders with extreme gaps). These two metrics are practically used to measure the user experience in the DiDi platform.

Table VII shows the online user experience of traditional EKF and our TCN model with different hyperparameters. We observe that, when the speed enhancement factor $\alpha = 6$ and the time attenuation factor $\delta = 20$, both gap rates and extreme gap rates achieve the best. Compared with the EKF, our TCN model reduces gap rates from 13.529% to 10.271% (↓24.01%) and extreme gap rates from 8.3% to 6.108% (↓26.4%).

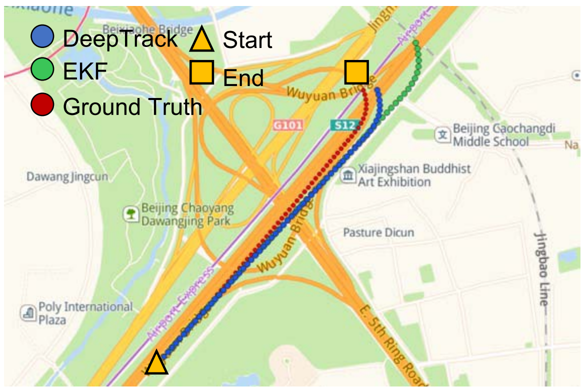
VIII. Related Work

A. Phone Pose Estimation

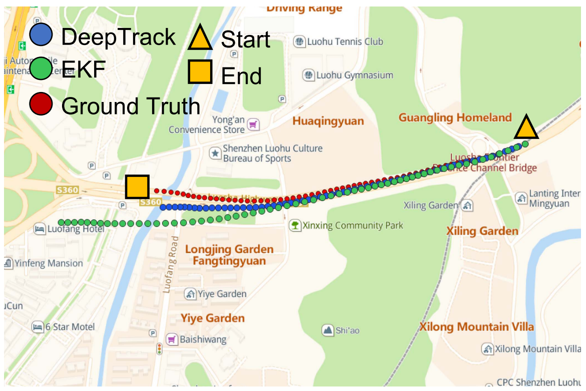
Since the phone's coordinate system is not always aligned with the vehicle, the inertial readings of the smartphone cannot be directly used to infer the vehicle's movement. There are many studies focusing on the phone's heading estimation for indoor localization, navigation, and tracking. Walkie-Markie [13] leverages the reading of the gyroscope to estimate the heading for indoor pathway mapping. Wang *et al.* [14] design the UnLoc system to employ the gyroscope and compass for accurate heading estimation in indoor dead-reckoning. Candan and Soken [15] propose two covariance-tuning methods to augment the Kalman filter (RKF) algorithm for IMU's attitude estimation. Another solution is to build a posture relationship between smartphones and vehicles. Wang *et al.* [16] leverage the embedded sensors to predict the smartphone's posture in car. Gao *et al.* [17] compared of the shadow tracking method and the 3-D tracking method.

B. Indoor Localization

At present, vehicle tracking services are usually based on the GNSS [18]. Drivers also expect to track and navigate their vehicles anywhere and anytime, even in underground or blocked environments [18], such as tunnels and overpasses. However, when driving into such scenarios, the satellite signal is in low intensity and cannot provide accurate locations, and inertial tracking errors accumulate to extreme values.



(a)



(b)

Fig. 20. Examples of real-time vehicle inertial tracking in (a) overpass in Beijing and (b) tunnel in Shenzhen.

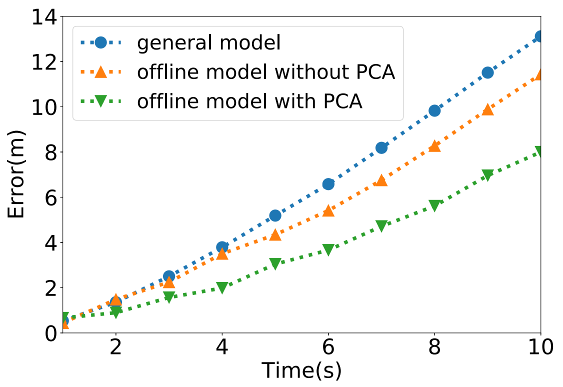


Fig. 21. Effectiveness of off-line customized learning (refinement with the dataset ②) and pose estimation with PCA. All results are tested in the dataset ②.

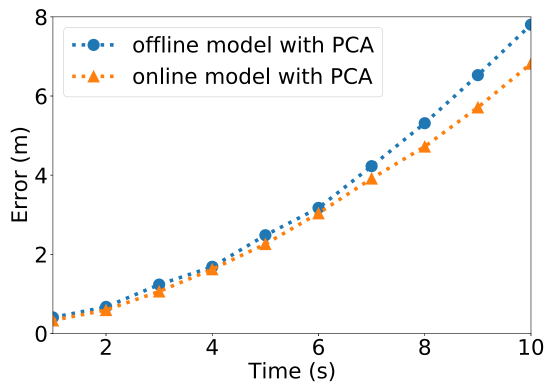


Fig. 22. Effectiveness of online customized learning (refinement by the latest online dataset ③). Both results are tested in the latest dataset ③.

Ultrasonic wave [19], Wi-Fi [20], and Bluetooth [21] technologies are mainly used for indoor localization. However, these solutions highly depend on dedicated deployments in the environment, e.g., users need to carry special devices for sensing the environment; thus, it constraints the wide deployment of indoor location-based services.

C. Inertial Tracking

The inertial dead reckoning has been widely used for indoor localization, e.g., step counting [22]. Mapcraft [23] uses the floor plan to reduce estimated errors of stride length and heading direction. You *et al.* propose a hybrid method based on IMU and RSSI [24] for pedestrian's dead reckoning. Hilsenbeck *et al.* [25] leverage Wi-Fi fingerprints to improve the tracking accuracy. Aghili and Su [26] present a robust 6-DOF relative tracking method by combining the noise-adaptive Kalman filter (AKF) and the inertial measurement unit (IMU). However, they all rely on the step counting results that are designed for pedestrians rather than vehicles. Gao *et al.* [17] propose VeTrack to track vehicles at low speeds in underground parking lots. It requires a large number of landmarks (e.g., bumps and turns) to calibrate a vehicle's location, but such landmarks are practically sparse in tunnels and overpasses. In addition, the inertial noises are always volatile and inconstant; thus, they are practically hard to eliminate and cause extreme error accumulation.

TABLE VII
USER EXPERIENCE WITH REAL-WORLD ORDERS (%)

Model	**Factor α**	**Factor δ**	**Gap rate**	**Extreme gap rate**
EKF	–	–	13.529	8.300
TCN	1	10	11.017	6.931
	1	20	10.875	6.738
	6	20	**10.271**	**6.108**

D. Sequential Deep Learning

RNN is a classic way to learn sequential data and has been widely used in natural language processing [27], machine translation [28], stream data processing [29], and traffic prediction [30]. LSTM [31] is a special RNN that controls the transmission state through gates. Recently, a convolutional neural network (CNN) [32] has achieved breakthroughs in learning sequence with better performance than RNN in learning long-term sequences. As a unique CNN, TCN [33] realizes the causal transmission of data by means of causal convolutions and expands the receptive field by dilated convolutions; thus, it is more flexible and efficient for mobile devices. In recent years, there has been explosive growth in applying deep neural networks for inertial tracking. IONet [34] abandons traditional steps counting and explores an inertial sequence learning approach (i.e., LSTM) to cure the curse of drift in IMU. Backprop KF [35] uses DNNs to extract effective features and feeds them to a predefined physical model to improve the filtering algorithms. Brossard *et al.* [36] combine the Kalman filter and deep neural networks to dynamically adjust filter parameters. Chen *et al.* present DynaNet [37], a hybrid deep learning and time-varying state-space model, which is trained end-to-end and performs excellently on a number of physically challenging tasks, including visual odometry, sensor fusion for visual-inertial navigation, and motion prediction.

IX. Conclusion

In this article, we conduct extensive experiments on real-world crowdsourced traffic datasets to explore the key factors of vehicle inertial tracking. We summarize three technical challenges and propose a novel vehicle inertial learning framework. It consists of a coordinate transformation algorithm, inertial sequence learning model, and customized retraining mechanism. The experimental results show our improvements in accuracy, robustness, and user experience of vehicle tracking in GPS-blocked environments.

References

[1] Y. Gao, Z. Yao, X. Cui, and M. Lu, “Analysing the orbit influence on multipath fading in global navigation satellite systems,” *IET Radar, Sonar Navigat.*, vol. 8, no. 1, pp. 65–70, Jan. 2014.

[2] P. Bahl and V. N. Padmanabhan, “Radar: An in-building RF-based user location and tracking system,” in *Proc. IEEE INFOCOM*, vol. 2, Mar. 2000, pp. 775–784.

[3] X. Wang, L. Gao, S. Mao, and S. Pandey, “CSI-based fingerprinting for indoor localization: A deep learning approach,” *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[4] F. Zafari, I. Papapanagiotou, and K. Christidis, “Microlocation for Internet-of-Things-equipped smart buildings,” *IEEE Internet Things J.*, vol. 3, no. 1, pp. 96–112, Feb. 2016.

[5] *iBeacon*. Accessed: Jul. 27, 2021. [Online]. Available: https://developer.apple.com/ibeacon/

[6] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, “Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards,” *Comput. Commun.*, vol. 30, no. 7, pp. 1655–1695, May 2007.

[7] S. Holm, “Hybrid ultrasound-RFID indoor positioning: Combining the best of both worlds,” in *Proc. IEEE Int. Conf. (RFID)*, Apr. 2009, pp. 155–162.

[8] Decawave. *Real Time Location: An Introduction*. Accessed: Jul. 27, 2021. [Online]. Available: http://www.decawave.com/sites/default/files/resources/aps003_dw1000_rtls_introduction.pdf

[9] J. P. Collins and R. B. Langley, “Mitigating tropospheric propagation delay errors in precise airborne GPS navigation,” in *Proc. Position, Location Navigat. Symp. (PLANS)*, Apr. 1996, pp. 582–589.

[10] N. Trawny and S. I. Roumeliotis, “Indirect Kalman filter for 3D attitude estimation,” Dept. Comput. Sci. Eng., Univ. Minnesota, Minneapolis, MN, USA, Tech. Rep. 2005-002, 2005, vol. 2.

[11] Y. Tong, S. Zhu, Q. Zhong, R. Gao, C. Li, and L. Liu, “Smartphone-based vehicle tracking without GPS: Experience and improvements,” in *Proc. IEEE ICPADS*, Mar. 2021, pp. 1–8.

[12] O. J. Woodman, “An introduction to inertial navigation,” Univ. Cambridge, Comput. Lab., Cambridge, U.K., Tech. Rep. 696, 2007.

[13] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, “Walkie-markie: Indoor pathway mapping made easy,” in *Proc. 10th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, 2013, pp. 85–98.

[14] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, “No need to war-drive: Unsupervised indoor localization,” in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2012, pp. 197–210.

[15] B. Candan and H. E. Soken, “Robust attitude estimation using IMU-only measurements,” *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–9, 2021.

[16] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, “Sensing vehicle dynamics for determining driver phone use,” in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2013, pp. 41–54.

[17] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, and G. Luo, “Smartphone-based real time vehicle tracking in indoor parking structures,” *IEEE Trans. Mobile Comput.*, vol. 16, no. 7, pp. 2023–2036, Jul. 2017.

[18] P. Carbone, D. Petri, and A. Tsourdos, “I&M in navigation systems [framing I&M topics],” *IEEE Instrum. Meas. Mag.*, vol. 18, no. 3, pp. 36–39, May 2015.

[19] F. Sato, Y. Motomura, C. Premachandra, and K. Kato, “Absolute positioning control of indoor flying robot using ultrasonic waves and verification system,” in *Proc. 16th Int. Conf. Control, Autom. Syst. (ICCAS)*, Oct. 2016, pp. 1600–1605.

[20] S. Zhang, W. Wang, and T. Jiang, “Wi-Fi-Inertial indoor pose estimation for microaerial vehicles,” *IEEE Trans. Ind. Electron.*, vol. 68, no. 5, pp. 4331–4340, May 2021.

[21] S. Tomažič and I. Škrjanc, “An automated indoor localization system for online Bluetooth signal strength modeling using visual-inertial SLAM,” *Sensors*, vol. 21, no. 8, p. 2857, Apr. 2021.

[22] Y. Shu, K. G. Shin, T. He, and J. Chen, “Last-mile navigation using smartphones,” in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2015, pp. 512–524.

[23] Z. Xiao, H. Wen, A. Markham, and N. Trigoni, “Lightweight map matching for indoor localisation using conditional random fields,” in *Proc. IPSN 13th Int. Symp. Inf. Process. Sensor Netw.*, Apr. 2014, pp. 131–142.

[24] Y. You and C. Wu, “Hybrid indoor positioning system for pedestrians with swinging arms based on smartphone IMU and RSSI of BLE,” *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–15, 2021.

[25] S. Hilsenbeck, D. Bobkov, G. Schroth, R. Huitl, and E. Steinbach, “Graph-based data fusion of pedometer and WiFi measurements for mobile indoor positioning,” in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Sep. 2014, pp. 147–158.

[26] F. Aghili and C.-Y. Su, “Robust relative navigation by integration of ICP and adaptive Kalman filter using laser scanner and IMU,” *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 4, pp. 2015–2026, Aug. 2016.

[27] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 160–167.

[28] A. M. Dai and Q. V. Le, “Semi-supervised sequence learning,” in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 3079–3087.

[29] T. Li, Z. Xu, J. Tang, and Y. Wang, “Model-free control for distributed stream data processing using deep reinforcement learning,” in *Proc. VLDB Endowment*, 2018, pp. 705–718.

[30] R. Gao *et al.*, “Aggressive driving saves more time? Multi-task learning for customized travel time estimation,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1689–1696.

[31] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proc. EMNLP*, 2014, pp. 1746–1751.

[33] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” 2018, *arXiv:1803.01271*.

[34] C. Chen, X. Lu, A. Markham, and N. Trigoni, “IONet: Learning to cure the curse of drift in inertial odometry,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 6468–6476.

[35] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, “Backprop KF: Learning discriminative deterministic state estimators,” in *Proc. Neural Inf. Process. Syst. (NIPS)*, vol. 2016, pp. 4376–4384.

[36] M. Brossard, A. Barrau, and S. Bonnabel, “AI-IMU dead-reckoning,” *IEEE Trans. Intell. Vehicles*, vol. 5, no. 4, pp. 585–595, Dec. 2020.

[37] C. Chen, C. X. Lu, B. Wang, N. Trigoni, and A. Markham, “DynaNet: Neural Kalman dynamical model for motion estimation and prediction,” 2019, *arXiv:1908.03918*.

Yao Tong received the B.S. degree in software engineering from Beijing Jiaotong University, Beijing, China, in 2019, where she is currently pursuing the M.S. degree in software engineering.

Her research interests include cellular network positioning and mobile computing.

Shuli Zhu received the B.S. and M.S. degrees in software engineering from Beijing Jiaotong University, Beijing, China, in 2019 and 2011, respectively, where he is currently pursuing the Ph.D. degree in software engineering.

His research interests include urban positioning and mobile computing.

Xiaotong Ren received the B.S. degree in software engineering from Beijing Jiaotong University, Beijing, China, in 2021, where she is currently pursuing the master's degree.

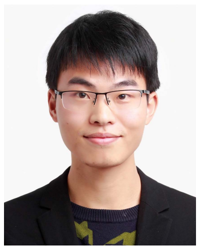
Her research interests include indoor pedestrian navigation and mobile computing.

Qinkun Zhong is currently pursuing the bachelor's degree in software engineering with Beijing Jiaotong University, Beijing, China, and will receive the degree in June 2022.

He is now looking forward to graduate study in North America. His research interests include simultaneous localization and mapping (SLAM), artificial intelligence, computer vision, and cryptography.

Dan Tao received the B.S. and M.S. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China, in 2001 and 2004, respectively, and the Ph.D. degree from the School of Computer Science, Beijing University of Posts and Telecommunication, Beijing, China, in 2007.

She was a Visiting Scholar with the Wireless Networking Laboratory, Illinois Institute of Technology, Chicago, IL, USA, from 2010 to 2011. She is currently a Professor with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing. Her research interests include wireless networks, the Internet of Things (IoT), mobile computing, and big data security.



Chi Li received the B.S. degree from Nanchang Hangkong University, Nanchang, China, in 2014, and the M.S. degree from Beihang University, Beijing, China, in 2017.

He is currently a Location Algorithm Engineer with DiDi Company, Beijing. His research interests include inertial navigation, integrated navigation, and machine learning.

Lei Liu received the B.S. degree from Shandong University, Jinan, China, in 2010, and the M.S. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2013.

He is currently an expert algorithm engineer with DiDi Company, Beijing, in charge of positioning and point of interest (POI) recommendation.

Ruipeng Gao received the B.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010, and the Ph.D. degree from Peking University, Beijing, in 2016.

He was a Visiting Scholar with Purdue University, West Lafayette, IN, USA, in 2019. He is currently an Associate Professor with the School of Software Engineering, Beijing Jiaotong University, Beijing. His research interests include mobile computing and applications, the Internet of Things, and intelligent transportation systems.