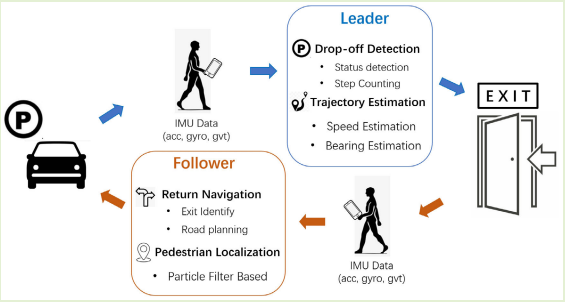
Help You Locate the Car: A Smartphone-Based Car-Finding System in Underground Parking Lot

Xiaotong Ren, Shuli Zhu, Feng Liu, Haitao Li, Haohang Li, Xuan Xiao, Ruipeng Gao, and Zhang Zhang

***Abstract*—While location awareness is common outdoors due to global navigation satellite system (GNSS) systems and devices, pedestrians are back into darkness indoors such as in underground parking lots. We often forget where we parked the car and get confused by such maze-like structure. In order to help drivers find their cars without any additional equipment and map support, we propose a car-finding navigation system that only relies on smartphones. It automatically identifies the user's drop-off point, records the walking trajectory from car's location to the exit, and provides fine-grained return navigation to help user back to the car. To address the accuracy and diversity of pedestrian tracking, we propose an inertial sequence learning framework based on outdoor crowdsourced trajectories, which avoids the dedicated efforts on groundtruth collection for model training. We also explore a smartphone posture detection method that supports multiple placements of the smartphone, and a support vector machines (SVM)-based trajectory refinement algorithm with only inertial readings. Besides, we explore a particle filter framework to track and navigate the pedestrian in real time. We have developed a prototype and conducted a series of experiments in multiple underground parking lots, and the results have demonstrated our effectiveness compared with the state-of-the-art.**



***Index Terms*—Indoor localization, inertial tracking, mobile crowdsensing, vehicle navigation.**

I. Introduction

NOWADAYS, driving has become a common way of travel, and frequently we drive into underground parking lots in shopping malls and transportation junctions (Fig. 1), where we may forget where we have parked in such complex structures. While it is easy to acquire outdoor locations with the global navigation satellite system (GNSS), it is difficult in GNSS-denied environments such as underground parking lots. Therefore, it is essential to record users' drop-off points and help them return to the car from an exit. Fortunately, smartphones capture plenty of data through multiple sensors (e.g., inertial measurement unit (IMU), WiFi, and Bluetooth), which could be exploited for real-time positioning and navigation.

Regarding pedestrian tracking, a straightforward method is using the Kalman filter [1], [2] over inertial readings. However, it causes unbounded tracking errors due to the accumulation of double integration. To overcome this issue, step counting [3], [4], [5], [6] techniques have been proposed which effectively prevent double accumulated errors. Nevertheless, determining an appropriate stride length beforehand for each user remains challenging. Another approach involves utilizing environmental sensing information as landmarks for localization, e.g., geomagnetic fields [7] and WiFi signatures [8], [9], but it is difficult to recognize such landmarks and construct the fingerprint dataset in underground parking lots due to the sparse coverage. Furthermore, there are already several IMU datasets (e.g., RIDI [10], RoNIN [11], and OxIDD [12]) designed for indoor scenes, but most of them rely on accurate calibration and require additional equipment for ground truth collection, preventing them for large-scale deployment.

Manuscript received 2 November 2023; revised 24 December 2023; accepted 24 December 2023. Date of publication 9 January 2024; date of current version 29 February 2024. This work was supported in part by NSFC under Grant 62072029, in part by the Beijing Nova Program, in part by Beijing NSF under Grant L221003, in part by the DiDi Research Collaboration Plan, and in part by the OPPO Research Fund. An earlier version of this paper was presented at the 18th International Conference on Mobility, Sensing and Networking (MSN), 2022 [DOI: 10.1109/MSN57253.2022.00122]. The associate editor coordinating the review of this article and approving it for publication was Prof. Fuqiang Gu. *(Corresponding authors: Ruipeng Gao; Zhang Zhang.)*

Xiaotong Ren, Shuli Zhu, Feng Liu, Haitao Li, Haohang Li, Xuan Xiao, and Ruipeng Gao are with the School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: xiaotren@bjtu.edu.cn; zhushuli@bjtu.edu.cn; liufeng@bjtu.edu.cn; haitaoli@bjtu.edu.cn; 20301132@bjtu.edu.cn; xiaoxuan@bjtu.edu.cn; rpgao@bjtu.edu.cn).

Zhang Zhang is with the China Institute of Electronic Technology Standardization, Beijing 101102, China (e-mail: zhangzhang@cesi.cn).

Digital Object Identifier 10.1109/JSEN.2024.3349385

Fig. 1. Example scenario for car finding. When pedestrians drive to the mall, they park their cars in the underground parking lot and walk inside the mall. When they leave the mall, they enter from the exit of the parking lot and walk back to their cars. Note that pedestrians frequently forget where their cars are parked, especially in large and complex underground parking lot.

To solve these problems, we propose a car-finding navigation system that only relies on smartphones. This innovative solution automatically detects the drop-off point and records the user's walking trajectory from there to the exit. After that, upon returning to the parking lot, it generates the user's previous path for efficient navigation toward the vehicle. Remarkably, this system only uses IMU data from smartphones and does not rely on any additional equipment or map collection, thus it can be easily deployed in any underground parking with zero efforts.

However, there exists three challenges during our exploration. First, it is difficult to gather accurate and sufficient ground truth data to train the indoor inertial tracking model without additional equipment. To address this issue, we devise an inertial sequence learning framework which is trained by outdoor global positioning system (GPS) trajectories, and it can be directly applied for indoor inference. Second, because the built-in IMUs in smartphones usually produce low-quality inertial readings that are prone to extreme noises. Therefore, we employ a bearing estimation method with posture detection, posture rotation, and drift error correction to mitigate the errors during tracking. Third, unlike vehicles that always ride following the designated lanes, pedestrian's movement exhibits randomness which diminishes the reliability of navigation. Thus, during the return period, we employ a particle filter approach to track the real time position and provide fine-grained navigation.

Our early work on indoor pedestrian tracking [13] was published in 2022. However, the early work can only realize the estimation of pedestrian trajectory, and do not include the guidance for pedestrians to find cars in reverse. In addition, we use more widely applicable smartphone posture recognition, data rotation methods, and new trajectory refinement methods in this study.

To sum up, our contributions are listed as follows.

1) We device an inertial sequence learning framework. It uses the IMU data collected outdoors to train the model with GPS data collected outdoors as groundtruth. In indoor scenarios, we only use IMU data for inference. In this way, the challenge of collecting groundtruth indoors is effectively addressed and our method can be applied in various underground parking lots.
2) A precise pedestrian bearing estimation method is proposed. The proposed approach utilizes accelerometer (acc), gyroscope (gyro), and gravity accelerometer (gvt) data to identify various smartphone postures and perform a rotation for the gyroscope data. Additionally, a trajectory drift error correction algorithm based on support vector machines (SVM) turning detection is employed to further reduce the influence of IMU data drift error.
3) A pedestrian locating method based on particle filter is proposed. The method utilizes the trajectory from the drop-off point to the exit to create a road-based constraint for updating the state of the particle. This approach results in a more precise prediction of the user's location.
4) We performed experiments in different scenarios to verify the performance of the method, compared it with other pedestrian tracking methods, and developed a prototype.

Specially, this article is organized as follows. In Section III, we introduce the overall framework and workflow of the car finding system. In Section IV, the inertial sequence learning framework is introduced, and the bearing estimation method is introduced in Section V. In Section VI, we introduce the pedestrian tracking method in the process of reverse car finding. In Section VII, we focus on the experimental results of the method, and give the conclusion in Section VIII.

II. Related Work

A. Basic Methods of Trajectory Estimation

The conventional approach for pedestrian tracking involves integrating the acceleration and angular velocity data of pedestrians to derive their speed and angle changes. However, due to the inherent noise in sensors, it is necessary to apply a Kalman filter [1] for filtering before integration. Aghili and Su [2] propose a robust 6-DOF relative tracking method that utilizes Kalman filter and IMUs. Nevertheless, this method still fails to completely eliminate sensor drift errors, which tend to be amplified after repeated integration. The step counting method [3], [4], [5] has emerged as a popular approach to mitigate sensor drift errors in recent years. By leveraging accelerometer data to detect pedestrian gait characteristics and estimate step length, it enables the calculation of walking distance. But due to the inherent challenge of accurately setting a universal step length for all pedestrians using this method, there exists an inevitable margin of error. To address this issue, Mapcraft [6] leverages floor plans to minimize estimated errors in both step length and heading direction estimation; however, this approach is contingent upon the availability of accurate maps.

B. Inertial Dead Reckoning

When it comes to the inertial dead reckoning, integration of gyroscope data is commonly employed. However, the utilization of gyroscopes often leads to significant errors. To address this issue, certain methods have incorporated

geomagnetic signals as an alternative. For instance, Muse [7] utilized geomagnetic signals for correcting gyro signals. Microsoft research introduced PathGuid [14], [15], an application that leverages distinct characteristics of geomagnetic signals at various locations within indoor environments as landmarks for recording and calibrating pedestrians' direction and position. UnLoc [16] not only utilizes geomagnetic field, but also considers specific patterns from WiFi and elevator accelerometers as landmarks for pedestrian localization purposes. Tong et al. [17] utilize principal component analysis [18] to infer the direction of walking in their study.

C. Indoor Pedestrian Trajectory Dataset

RIDI [10] is the pioneering publicly available dataset for indoor pedestrian tracking, including IMU data and corresponding groundtruth. The IMU data are gathered using ordinary mobile phones, while the groundtruth is obtained through Google Tango mobile phones. Additionally, RIDI stands as the first data-driven inertial navigation method that computes pedestrian speed in the mobile coordinate system and employs a multi-source fusion approach to estimate heading. As an enhancement, Yan et al. [11] introduce the RoNIN dataset which incorporates meticulous precalibration and postcalibration procedures for each trajectory to minimize errors arising from disparate equipment used during data and groundtruth acquisition. At the same time, RoNIN employs a depth model consisting of three different cores (long short-term memory (LSTM), temporal convolutional network (TCN), and ResNet) to extract pedestrian trajectories. ADVIO [19] utilizes an iPhone 6s as an IMU collection device to provide pseudo groundtruth generated by their handcrafted inertial odometry algorithm. However, the dataset itself is limited with only 23 sequences available. On the other hand, OxIDD [12] encompasses various acquisition postures and walking speeds with groundtruth collected using Vicon technology, resulting in a dataset of 158 sequences. Nevertheless, all of above methods require additional equipment for data collection and are restricted to fixed scenes where the necessary devices are installed, thus rendering them unsuitable for our scenario. In this article, we proposes a method that solely relies on users' mobile phones for collecting IMUs data and groundtruth.

D. Wifi-Based Indoor Positioning Method

Using WiFi for positioning is a widely adopted approach in the field of indoor positioning. Hilsenbeck et al. [20] proposed the Hapi system, which utilizes off-the-shelf standard WiFi technology to provide pseudo-3D indoor positioning. Bell et al. [8] exploit WiFi fingerprints to enhance tracking accuracy effectively. Moreover, Lee et al. [9] deployed a WiFi positioning system on smartwatches, demonstrating improved execution time and accuracy through location awareness learning based on random forest algorithm. Additionally, WiDeep [21] presents an innovative method that combines depth models with WiFi signals for enhanced performance in indoor positioning applications. The system is capable of effectively mitigating the noise present in the received WiFi signal and accurately capturing the intricate correlation between the signals emitted by WiFi access points (APs) and the mobile phone's location. However, relying solely on WiFi for pedestrian tracking and positioning within indoor underground parking lots proves unfeasible due to significantly weak WiFi signal coverage throughout most areas within such facilities. Generally, only at the entrance and exit points can relatively stronger WiFi signals be detected.

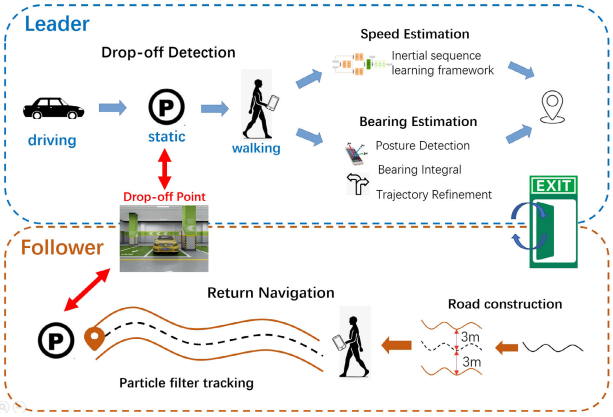


Fig. 2. System overview. It consists of the leader phase and the follower phase. In leader phase, we record the users' trajectory from the drop-off point to the exit, while in follower phase, our aim is to provide fine-grained navigation for users helping them return to their cars.

III. Overview

As illustrated in Fig. 2, the pipeline of our method consists of two parts, namely the leader phase and the follower phase. The leader's objective is to record the trajectory from the drop-off point to the exit. Initially, we perform drop-off point detection to accurately identify when the user gets off. Subsequently, while the pedestrian walks toward the exit, we continuously track and record the corresponding trajectory. In the follower phase, our aim is to provide navigation for users, helping them return to their cars. Specially, we assume that the exit remains consistent and generate a planned route for navigation assistance. Concurrently, we employ real-time tracking techniques with particle filter to estimate and update the user's location dynamically.

The leader phase contains two steps, i.e., drop-off detection and pedestrian tracking. For the drop-off point detection, our basic idea is that when the user's status changes from driving to static and further to walking, there is a drop-off point. Thus, we define user's status as {driving, walking, static}, and design the process of drop-off point detection as Fig. 3 shown. We use the triaxial combined acceleration value without gravity acceleration to identify the user's current status. Additionally, to mitigate potential misidentification caused by smartphone's jitter, we incorporate a robust step counting algorithm. Step counting uses IMU data and relies on the combined acceleration of the triaxial acceleration. Since the triaxial acceleration of IMU will show regular fluctuations with time when pedestrian is walking, it is possible to judge whether a pedestrian is walking and count the number of steps through the change pattern of the peak and trough values in a time window. Specifically, when observing the continuous

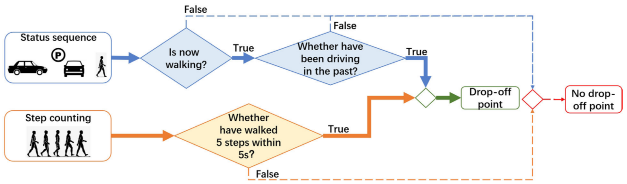


Fig. 3. Workflow of drop-off point detection. When observing the continuous increase of the number of steps while the user is walking, we confidently determine that user has disembarked.

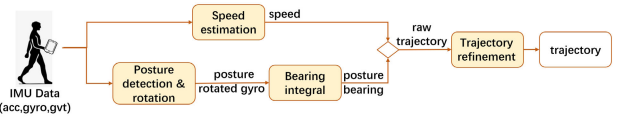


Fig. 4. Workflow of pedestrian tracking. To estimate walking speed, we employ an LSTM-based model to infer changes in pedestrian indoor velocity.

increase of the number of steps during walking, we confidently determine that user has disembarked.

Next, the pedestrian tracking process commences. We address speed estimation and bearing estimation as separate tasks, following the workflow described in Fig. 4. To estimate speed, we employ an LSTM-based model trained on outdoor IMU and GPS data, utilizing only accelerometer and gyroscope readings from smartphones to infer changes in pedestrian velocity per second indoors. For bearing estimation, a SVM-based model is employed to detect smartphone posture, enabling us to appropriately adjust gyro data since different postures yield distinct representations of the user's walking motion. Then, the gyroscope integral method is utilized to calculate the change in pedestrian bearing per second, enabling us to combine speed and bearing change for trajectory estimation. However, as mentioned above, traditional integration methods and relatively cheap IMU sensors may cause significant drift errors. To address this issue, we propose a postprocessing approach for error drift correction. The fundamental idea involves categorizing the user's walking behavior into two types: right angle turns and straight paths. We employ an SVM-based turn detection method to identify turning points in the trajectory. Subsequently, any bearing changes observed during straight sections are considered as drifts and corrected accordingly.

When the user intends to return to his car, we proceed to the follower phase, whose workflow is described in Fig. 5. We first generate the planned route by utilizing the pedestrian trajectory stored in the leader phase. As shown in Fig. 6, we first flip the trajectory, and then expand the road backbone by 3 m on both sides as the boundary of the road. Subsequently, the IMU data is fed into the same speed model and integration method employed in the pedestrian tracking module to derive the user's speed and bearing change per second. These speed and bearing changes are then utilized as influencing factors within a particle filter framework, with constraints imposed by the planned road boundary for updating particle status. The result of particle position serves as an accurate estimation of pedestrian location. Additionally, we extract turn point information from the trajectory and provide users with timely turn notifications when approaching these points.

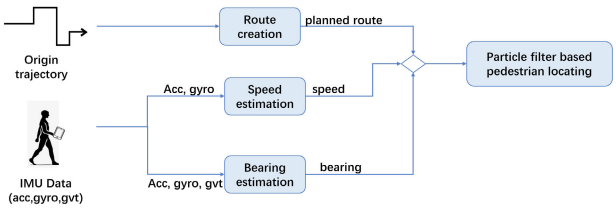


Fig. 5. Workflow of follower phase. We construct the planning road for navigation. For more precise localization, we use a particle filter framework to track the user's real time location.

Fig. 6. Planning road construction process. We first flip the trajectory stored in the leader phase, and then expand the road backbone by 3 m on both sides as the boundary of the road.

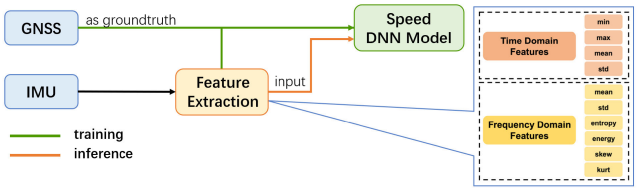


Fig. 7. Structure of inertial sequence learning framework. The speed deep-learning neural network (DNN) model is trained by outdoor data with GNSS data as groudtruth, and only use inertial data for indoor inference. Instead of using raw data directly, we extract feature of inertial data first.

IV. Inertial Sequence Learning Framework

For trajectory estimation, we adopt the scheme of PeTrack [13], which divides user trajectory calculation into two parts: speed estimation and bearing estimation. And in order to obtain the speed without high cumulative error, we do not use simple integration of accelerometer but use the deep learning model to estimate. However, in underground parking lot, it is difficult to obtain the groundtruth required for training the model by relying on smartphones alone. Therefore, we propose a framework where the model is trained in outdoor scenes and used indoors, because in outdoor scenes, smartphone can collect the IMU data with high accuracy GPS data which can be used as the groundtruth.

Fig. 7 shows the structure of the framework. Because the GPS signal is acquired at a frequency of 1 Hz, whereas the IMU data are acquired at a higher frequency of 50 Hz, we consider the IMU data sampled at 50 Hz along with the corresponding GPS data within each second. And since GPS provides the speed field, we utilize GPS_{Spd} as the groundtruth. Besides, instead of directly employing raw IMU data as input, we extract features from the raw data. These feature data including both time domain and frequency domain characteristics. In the time domain, we consider maximum, minimum, mean, and standard deviation as key features. In the frequency domain, we derive features from a spectral graph solved through fast Fourier transform (FFT). Specifically, these FFT-based features include mean value, standard deviation, information entropy, energy content, skewness indicating statistical data asymmetry direction and degree of distribution skewness, and kurtosis representing peak sharpness in the dataset.

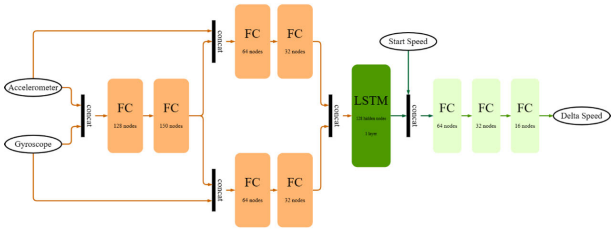


Fig. 8. Speed model structure. The input is acceleration and gyroscope and output is the change of speed per second. The core of speed model is LSTM.

We characterize the velocity model as follows:

$$\Delta \mathrm{Spd}_t = \mathrm{model}(\mathrm{Acc}_{t-1,t}, \mathrm{Gyro}_{t-1,t}, \mathrm{InitSpd}_t) \tag{1}$$

where $t = 1, 2, 3, \ldots, n$, $\mathrm{Acc}_{t-1,t}$ and $\mathrm{Gyro}_{t-1,t}$ represent the extracted accelerometer and gyroscope features from this second, respectively. Moreover, $\mathrm{InitSpd}_t$ denotes the initial speed at time t, and $\Delta\mathrm{Spd}_t$ represents the model's output indicating the change of walking speed during this second compared to $\mathrm{InitSpd}_t$.

We derive $\mathrm{InitSpd}_t$ through the following procedure:

$$\mathrm{InitSpd}_t = \begin{cases} \mathrm{GPS\_Spd}_t, & t \geq 1, \ \text{training} \\ 0.81, & t = 1, \ \text{inference} \\ \mathrm{Spd}_{t-1}, & t > 1, \ \text{inference}. \end{cases} \tag{2}$$

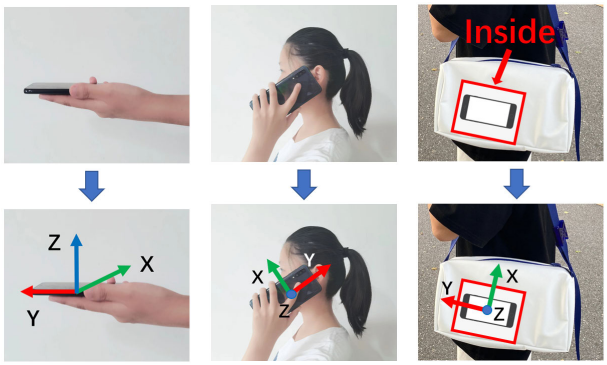
Notice that the method for obtaining $\mathrm{InitSpd}_t$ differs between training and inference. During training, we directly utilize the GPS speed of $t-1$ second as $\mathrm{InitSpd}_t$. However, during inference, due to the absence of GNSS signal, we adopt the median value (0.81 m/s) of average pedestrian speeds for $\mathrm{InitSpd}_t$ at $t = 1$. Because the speed estimation is only performed after the detection of the drop-off point, the pedestrian is already in the walking state, not static state. We evaluated the average walking speed of the first 3 s of each track in our dataset collected by 20 participants and finally determined the value of 0.81 m/s. For subsequent seconds ($t > 1$), we employ the estimated speed from the previous second (Spd_{t-1}) as $\mathrm{InitSpd}_t$.

The structure of the speed estimation model is illustrated in Fig. 8, which comprises three main components: embedding layer, representation layer, and regression layer. In the embedding layer, we initially extract the mixed embeddings of $\mathrm{Acc}_{t-1,t}$ and $\mathrm{Gyro}_{t-1,t}$ using a two-layer fully connected (FC) network. Subsequently, these embeddings are concatenated with $\mathrm{Acc}_{t-1,t}$ and $\mathrm{Gyro}_{t-1,t}$, respectively, to further enhance feature extraction. For the representation layer, we employ LSTM due to its suitability for our learning task of continuously estimating pedestrian speed from sequential IMU data. The structure is designed with one stack layer and a hidden layer dimension of 128. Finally, in the regression layer, we utilize four FC layers where the input data consists of not only the output from the representation layer but also $\mathrm{InitSpd}_t$.

Then we get Spd_t by

$$\mathrm{Spd}_t = \mathrm{InitSpd}_t + \Delta\mathrm{Spd}. \tag{3}$$

In the model training process, the sequence length is 50 samples/s, and total training set contains over 15 h of data.



(a) (b) (c)

Fig. 9. Smartphones and corresponding coordinate system with different postures. The coordinate system of smartphone is different at different posture. The flat posture is the standard coordinate in this article. (a) Flat. (b) Calling. (c) Bag.

We used SmoothL1Loss function and Adam optimizer, set the epochs to 80, the initial learning rate to 0.001, and the learning rate to decrease by 50% for every ten rounds of training.

V. Pedestrian Bearing Estimation

This section consists of three parts, smartphone posture detection, bearing calculate, and trajectory refinement.

A. Posture Detection

At present, our method is designed for smartphones of Android system. Android smartphones adopt a right-handed coordinate system, as shown in Fig. 9(a). In real-life scenarios, it is highly improbable for users to maintain the standard flat posture of the device [Fig. 9(a)], but rather they are more likely to engage in activities such as making phone calls while holding the device [Fig. 9(c)] or placing it into a bag [Fig. 9(c)]. Consequently, when the smartphone assumes different positions, the user's walking motion exhibits distinct variations along the three axes of the IMU. For instance, with the flat posture of smartphone, the user's forward acceleration is predominantly manifested along the Y-axis of the phone. Conversely, when the device is held for calling purposes, it is expressed on both the X- and Y-axis. This variation in feature expression can impact the efficacy of subsequent models introduced in this article. Therefore, a real-time posture detection module becomes imperative to ascertain the current orientation of the smartphone and accordingly match it with an appropriate training model for that specific posture. Currently, our method supports three smartphone postures, flat, calling, and cross bag positions as they are frequently encountered in daily life.

We take acc, gvt, and gyro data within a period of time window as the original input, extracted their time-domain features, including max/min/mean/median/std, and input them into the model. The window size can fluctuate within 1–5 s. The larger the window, the better the effect. The posture detection module is implemented using SVM and sklearn framework on Python. The parameters are set as follows: $C = 0.4$, kernel = "linear," decision_function_shape = "ovr." On the mobile side, we use the smile library to implement.

B. Bearing Calculation

Due to the existence of different postures of smartphone, we need to transform the gyroscope data from arbitrary posture ordinate to flat posture ordinate. First, the Z-axis in the standard coordinate system Z_{flat} is calculated by

$$Z_{\text{flat}} = -\frac{\text{gvt}}{\|\text{gvt}\|_2} \tag{4}$$

where gvt is the sum vector of the triaxial gravity

$$\text{gvt} = (\text{gvt}_x, \text{gvt}_y, \text{gvt}_z). \tag{5}$$

Second, the positive direction of Y-axis in the standard coordinate system Y_{flat}, is calculated by

$$Y_{\text{flat}} = \frac{\text{acc}_{\text{pdstr}}}{\|\text{acc}_{\text{pdstr}}\|_2} \tag{6}$$

where $\text{acc}_{\text{pdstr}}$ is the forward acceleration of the pedestrian. It is calculated by

$$\text{acc}_{\text{pdstr}} = (\text{acc}_x - \text{gvt}_x, \text{acc}_y - \text{gvt}_y, \text{acc}_z - \text{gvt}_z) \tag{7}$$

where acc_x, acc_y, acc_z is the triaxial acceleration. When we have the Y- and Z-axes, we cross the two axes to get the X-axis X_{flat}

$$X_{\text{flat}} = Y_{\text{flat}} \times Z_{\text{flat}}. \tag{8}$$

Then, the standard coordinate system is established. After that, the gyroscope data are projected into the standard coordinate system as follows:

$$\text{gyro}_{\text{flat}} = \begin{bmatrix} X_{\text{flat}} \\ Y_{\text{flat}} \\ Z_{\text{flat}} \end{bmatrix} \cdot \begin{bmatrix} \text{gyro}_x \\ \text{gyro}_y \\ \text{gyro}_z \end{bmatrix} \tag{9}$$

where $\text{gyro}_{\text{flat}}$ is the gyroscope in the standard coordinate, and gyro_x, gyro_y, gyro_z is the triaxial gyroscope of arbitrary posture. Finally, the projected gyroscope data is integrated to obtain the bearing changes.

C. Trajectory Refinement

As mentioned above, we obtain the speed of pedestrian from speed estimation model and obtain the bearing from bearing estimation, so that we can obtain the trajectory of pedestrian. But, there is still large drift error of the bearing we have obtained. Therefore, we add a postprocessing for refinement.

In this article, it is assumed that the user only goes straight and turns right angles. Then, in the straight segment, the bearing of the user should be the same, but in fact, it is always found that the bearing of the straight segment also changes as shown in Fig. 10 where brng_{i0} is the unit bearing of Seg_i. The reason is that since the built-in IMUs of smartphone are generally cheap, the collected data itself have a large error, which leads to a certain drift error between the calculated bearing and the groundtruth. Fortunately, the gyroscope drift amount is relatively constant in each posture and can be corrected as a whole. Based on this premise, the turning detection model is adopted to cut the straight sections of the trajectory by detected turnings, so as to correct the drift of the straight section. Since the feature expression of gyro data under different posture of smartphone is obviously different,

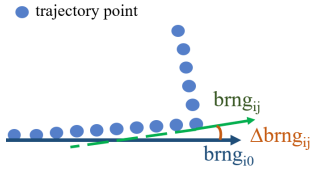


Fig. 10. Bearing drift of straight trajectory segment. Even if the pedestrian is walking in a straight line, the bearing will gradually deviate from the original direction.

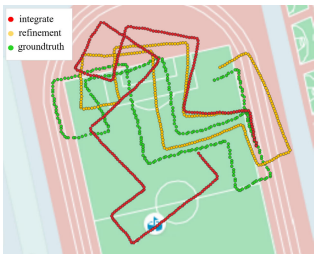


Fig. 11. Drift correction example. By using drift correction, the trajectory is closer to the groundtruth than the raw trajectory.

the model is trained first through the multiposture dataset collected outdoors. Each posture trains an independent turning detection model, which is essentially a classifier for classifying the current sample as turning or straight section. We used SVM to implement the model with decision_function_shape = "ovr." The input is gyroscope data within a period of time after noise reduction and smoothing. The window size is currently set to 3 s and the step size to 1. In practice, the current posture of the smartphone is obtained through the above-mentioned posture detection module, and the corresponding turning detection model is invoked for inference. In addition, in order to prevent errors caused by multiple detection of a single turn, all turns detected within 5 s are regarded as the same turn.

After obtaining the turn detection result of the trajectory, the trajectory is segmented according to turning points, and the trajectory between every two turning points is regarded as a segment. Thus, we obtain $\text{Seg}_i, i = 1, 2, \ldots, n$. Therefore, the bearing changes of each Seg_i will be counted as follows:

$$\Delta\text{brng}_i = \sum_{j=0}^{m} (\text{brng}_{ij} - \text{brng}_{i0}) \tag{10}$$

where Δbrng_i means total bearing change of Seg_i, m means that there are m trajectory points in this segment (1 point per second), and brng_{i0} is the init bearing of Seg_i. Then, the mean value of Δbrng_i will be calculated as the drift amount of the current Seg_i, which will be used for correction

$$\text{correct\_brng}_j = \text{brng}_{ij} - \Delta\text{brng}_i \div m, \quad j = 0, 1, \ldots, m. \tag{11}$$

As shown in Fig. 11, the trajectory after drift correction has lower error.

VI. Pedestrian Tracking

When the user is returning to car, we need to locate the user in real time, so as to better guide the user. We use a particle filter to estimate the pedestrian location. First, IMU data are input into the trajectory calculation part to obtain the current

speed and bearing of the pedestrian. Meanwhile, the planned path created as shown in Fig. 6 is used as the constraint of the particle filter to calibrate the pedestrian position in real time, which can effectively reduce the uncertainty of the pedestrian trajectory.

We represent the pedestrian position state as a probability distribution and use particle filter to update the pedestrian status. At time t, the pedestrian's state (position and heading) is expressed as $s(t)$ using multidimensional random variables, and each group of possible state values is defined as a particle. J particle set $\{s_t^{(j)}\}_{j=1}^{J}$ is used to represent possible distribution state of pedestrians of time t. In this framework, the algorithm follows the discrete time $\{1, \ldots, t-1, t, \ldots\}$ running and repeating the following three steps at each time. Without loss of generality, we assume that we have obtained J particles at time $t-1$ as $\{s_{t-1}^{(j)}\}_{j=1}^{J}$.

The status update module according to the known inputs $\{s_{t-1}^{(j)}\}_{j=1}^{J}$, the speed and bearing as the latest motion m_t, and forecast t moment condition $\{\hat{s}_t^{(j)}\}_{j=1}^{J}$. In order to capture the uncertainty of the motion and previous state, we add the random noise into the tracking module.

The weight updating module determines whether the particle is in the road, and then adjusts the weight of the particle $\{w_t^{(j)}\}_{j=1}^{J}$. If the particle is judged to be outside the road, multiply the weight of the particle by a minimum value $1e-8$. If the particle is still in the road, we do not modify its weight. After filtering out the particles of too small weight, the weight of the remaining particles is normalized to ensure that

$$\sum_{j=1}^{J} w_t^{(j)} = 1. \tag{12}$$

When the number of available particles is too small, the resampling module reselects and adds particles according to the weight distribution $\{w_t^{(j)}\}_{j=1}^{J}$ of the current state set $\{\hat{s}_t^{(j)}\}_{j=1}^{J}$, and then generates a new state set $\{s_t^{(j)}\}_{j=1}^{J}$ to replace the old state set $\{s_{t-1}^{(j)}\}_{j=1}^{J}$. The above three steps will be repeat during whole return navigation period. We set the max number of particle to 500. Once the number of available particle lower than 300, do resampling.

The pedestrian's final position at time t is

$$\text{position}_t = \sum_{j=1}^{J}\left(w_t^{(j)} \times s_t^{(j)}\right). \tag{13}$$

In addition, in the follower phase, not only the generated planned road can give the user the overall direction guidance, but also saved turn detection results of the trajectory refinement part can offer the corresponding turn guidance to the user.

VII. Evaluation

A. Drop-Off Point Detection

We chose an underground parking lot of a railway station in Hohhot, Inner Mongolia, China for the experiment.

TABLE I
Drop-Off Point Detection Experience Result

Number of tests	Number of successes	Success rate	Latency
19	17	89.5%	4.2s

TABLE II
Training Dataset for Speed Model

Scenarios	Posture	User	Number of trajectory
outdoor	flat	18	257
	calling	16	119
	bag	10	100

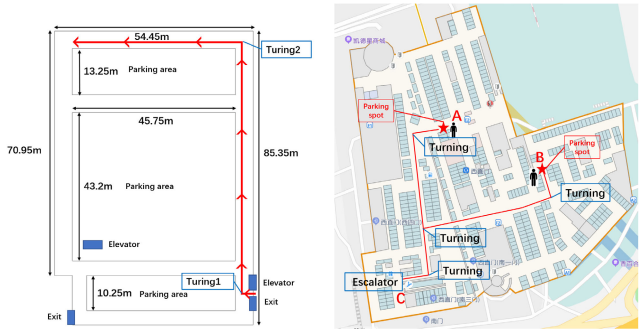
In the experiment, every time we drove into the parking lot from outside, we parked in the Taxi pick-up and drop-off area and got off the car to walk for a distance (>10 m). The experiment was carried out 19×. Table I shows the experimental results. It should be noted that due to the sensitive detection algorithm, the main failure scenarios are misidentification as walking and wrong step counting caused by different driving habits or road conditions (such as unstable driving and bumpy road surface) during driving.

B. Pedestrian Trajectory Estimation

Experiments of pedestrian trajectory estimation were conducted on two datasets collected by ourselves: the underground parking lot of a campus with an area of about 4660 m^2, and the underground parking lot of a large shopping mall with an area of about 24 600 m^2. By default, the experiment in this part adopts a single posture acquisition for each trajectory, and there is no posture transformation in the trajectory. The speed model in this experiment is trained using the data collected outdoors. The detailed information is shown in Table II. In total, we collected more than 20 h data (including training set and test set), by using five Android phones, since we have not yet found a large public dataset suitable for this scenario. We collected the GPS data as groundtruth outdoors to train the model. Indoors, we manually measured the coordinates of each turning point relative to the starting point. In the experiment, each participant in the experiment will have another participant record his turn time while walking, and then calculate the error between the pedestrian coordinates estimated by the model and that measured manually.

The map and track route of the campus underground parking lot are shown in Fig. 12(a). The length of a single track is about 123 m, including two turnings. And the map and track route of the shopping mall underground parking lot are shown in Fig. 12(b). We designed two routes, the first from A to C, about 140 m, including two turnings, the second from B to C, about 160 m, including three turnings. Table III lists the dataset details.

We respectively compared the trajectory estimation results on the three postures. In order to demonstrate the necessity of trajectory refinement algorithm, we compare the result of with and without trajectory refinement. Fig. 13 shows the example of the tracking result on different postures. Our method projects the gyroscope data and then makes integral, finally does the trajectory refinement base on turning



(a) (b)

Fig. 12. Map and test paths of the campus and large shopping mall underground parking lot. In campus underground parking lot, we design a route with two turnings. And in large shopping mall underground parking lot, we design a route with two turnings, and another route with three turnings. (a) Campus. (b) Shopping mall.

TABLE III
TEST DATASET FOR PEDESTRIAN TRAJECTORY ESTIMATION

Scenarios	Posture	User	trajectory
campus underground parking lot	flat	4	16
	calling	8	32
	bag	8	32
shopping mall underground parking lot	flat	8	64
	calling	4	32
	bag	4	32

detection. On the three postures, the refinement algorithm has significantly optimized the trajectory. Fig. 14 shows the detailed error value per hundred meters in campus parking lot, and Fig. 15 shows the detailed error value per hundred meters in shopping mall parking lot.

Fig. 16 shows the final point error of the trajectory of the proposed method in different postures. It can be seen that the result under flat posture is the best. In calling and bag posture, due to the drift error brought by the posture, the result has decreased to a certain extent, but the maximum error is not more than 10 m.

C. Comparison of Baseline Tracking Methods

In order to verify the applicability and superiority of the pedestrian trajectory estimation method proposed for the application scenario of this article, we compare our method with other algorithms in related fields. Note that all comparison experiments are performed on the dataset of the hand-held flat posture.

First, the distance estimation results of the speed model are compared with the traditional integral method and step counting algorithm, and Fig. 17 shows the results. All the data used in this experiment are outdoor, and the groundtruth are GPS latitude and longitude. The test samples are a total of 20 tracks of four users. It should be noted that, since the purpose of this experiment is to compare the effect of speed calculation algorithms, the GPS is used to provide the bearing, rather than the bearing estimation method proposed in this article. And in order to prevent the result of integration from being too bad, we carry out denoising during data processing. It can be seen that our method has better results. Although we perform the denoising, the error of integration method is still largest. Integration still produce a large cumulative error. And the step counting algorithm uses fixed step size prior, which is difficult to fit situation of every pedestrian, causing the error.

In addition, we also added the comparison of duration. Our method takes an average of 0.0005 s to estimate the speed, while the integration method takes 0.0003 s. Although the integral method is faster, it has a large error. Our method takes more time, but it is still sufficient for use.

Second, the bearing estimation method with drift error correction in this article is compared with the pure gyro integral method, the method solely using geomagnetic field, and Muse which is the multisource fusion method. Since the purpose of this experiment is to compare the effect of bearing estimation method, the groundtruth of speed from GPS is used in the experiment, instead of the estimated speed by the method proposed in this article. As Fig. 18 shown, the method of pure gyroscope integration has the largest error due to the large angle offset generated by gyroscope drift. The MUSE algorithm introduced geomagnetic correction to the bearing, which was better than the pure gyroscope integration method. The geomagnetic method has higher accuracy, but there are usually complex magnetic fields in the indoor environment, which will have a great impact on the bearing estimation. The errors of the SVM-based drift correction algorithm proposed in this article is smaller than others.

In addition, we also compared the overall pedestrian tracking method in this article with RoNIN method, and the results are shown in Fig. 19. RoNIN algorithm has a large error in comparison, mainly because it requires Tango phone to collect groundtruth, accurate data calibration, and 200 Hz sampling frequency, which cannot be achieved in the target usage scenario in this article. The purpose of this article is to develop a car finding algorithm that can be used by the general public only with ordinary smartphone. In order to meet the usage scenario of this article, we reduce the frequency of the dataset provided by RoNIN author to 50 Hz, then the network is trained, and the data collected in this article (without accurate data calibration) is input into the network for trajectory estimation, and the final result is obtained. Therefore, the effect of RoNIN algorithm is much worse.

D. Long-Distance Tracking

In order to verify the effect of the algorithm in the long-distance scene, we further collect the long-distance data in the outdoor scene for the experiment. A total of 18 tracks are used, including three users. Fig. 20 shows the change of 100-m error of the proposed method with the change of trajectory length at different postures. It can be seen that the error per 100 m increases significantly as the length of the track increases. When the trajectory length increases to 500 m, even in flat posture, 95% error reaches 34.45 m, and the algorithm is basically unavailable. This indicates that the algorithm proposed is more effective on short distance tracking ($\leq$200 m), but has limited effect on long distance tracking. This is also an optimization method for our future. However, the distance of 200 m can basically cover most of the parking scenes of underground parking lots in daily life.

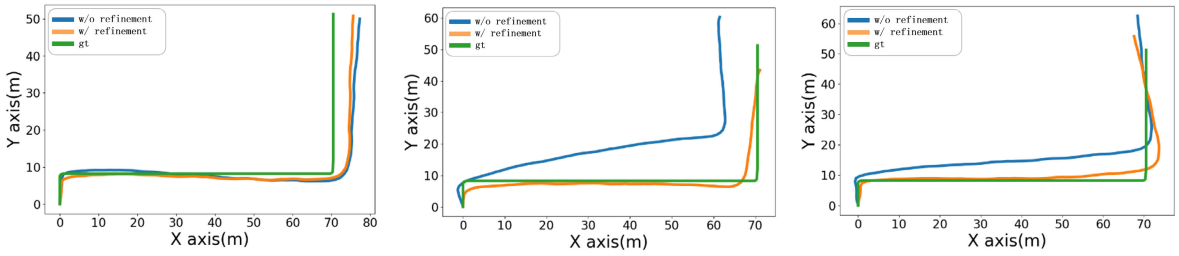


Fig. 13. Example of pedestrian tracking result with or without trajectory refinement in campus underground parking lot. Our method that first integrates the gyroscope and then corrects the drift has the better result. (a) Flat. (b) Calling. (c) Bag.

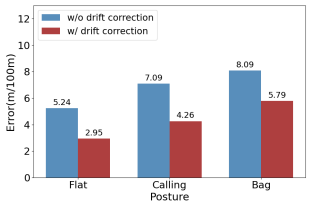


Fig. 14. Mean tracking error per 100 m in campus parking lot. After refinement, the error has reduced significantly on each posture.

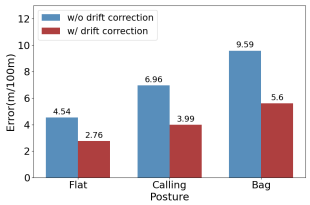


Fig. 15. Mean tracking error per 100 m in shopping mall parking lot. After refinement, the error has reduced significantly on each posture.

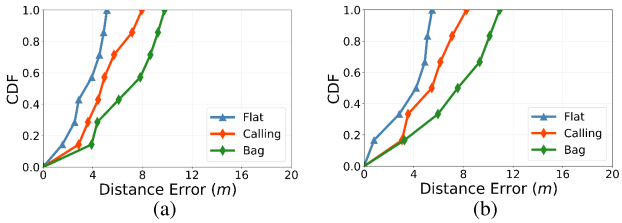


Fig. 16. Error of final point of trajectory in the campus and shopping mall underground parking lot with different postures. The result of flat posture is the best, and the result of bag posture is the worst because there is large drift in bag. (a) Campus. (b) Shopping mall.

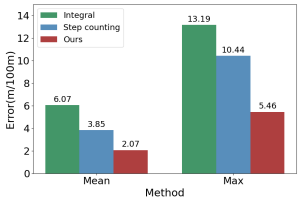


Fig. 17. Comparison of distance estimation result between our method and others. Our method has less error, because the integration method will produce a large cumulative error and the step counting algorithm uses fixed step size prior, which is difficult to fit situation of every pedestrian.

E. Return Navigation

We also carried out return navigation experiment on the above two datasets of campus parking lot and underground parking lot of large shopping malls.

Fig. 21 shows the comparison between the user trajectory obtained solely by using trajectory estimation method and the

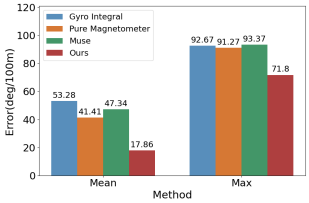


Fig. 18. Comparison of bearing estimation result between our method and other methods. Our method has less error because our method correct the drift error of trajectory after gyroscope integral.

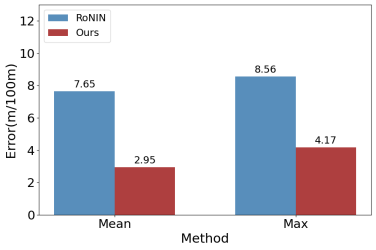


Fig. 19. Comparison of trajectory estimation result between our method and RoNIN. Our method has less error because RoNIN require strict calibration which is not supported in our scenes.

effect after adding particle filter on the campus parking lot dataset. It can be seen that the particle filter method has better effect on each posture.

Fig. 22 shows the final point error of the proposed method in different posture. Fig. 22(a) is the result of the dataset in campus parking lot and Fig. 22(b) is the result of the dataset in shopping mall underground parking lot. It can be seen that the result under flat posture is the best. In calling and bag posture, due to the drift error brought by the posture, the effect has a certain decline. However, no matter in the small-scale scene or the large-scale parking lot, the maximum error of each posture is less than 11 m.

In the application scenario, it is likely that the user has different postures leaving and back to the drop-off point. For example, the user makes a phone call when he leaving from the car and put the phone flat in his hand when he returns because he need to check the navigation from smartphone. Therefore, we conduct experiment on scenarios with different back-and-forth postures.

In this experiment, we assume that when the user leaves from drop-off point, the smartphone may be in any of the poses flat, calling and in the pocket, while when the user returns, the smartphone is flat in the hand, because in our application scenario, the user mostly needs to check the navigation on the smartphone. Table IV shows the results of the experiment

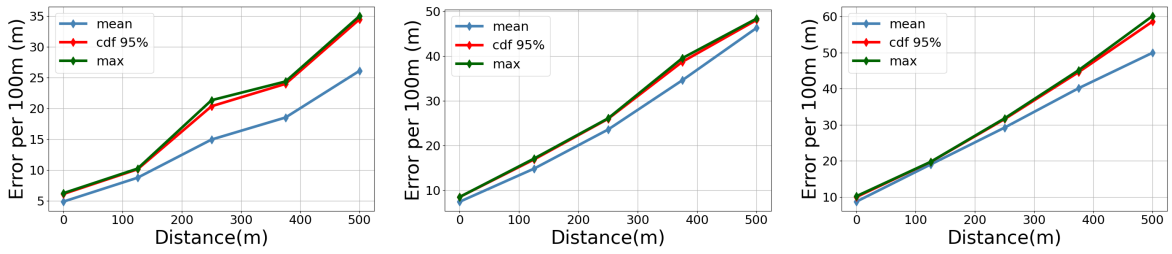


Fig. 20. Tracking error of long distance data in different postures. Our method on trajectories less than 200 m has the low error. But for trajectories longer than 200 m, the error is rapidly increase. (a) Flat. (b) Calling. (c) Bag.

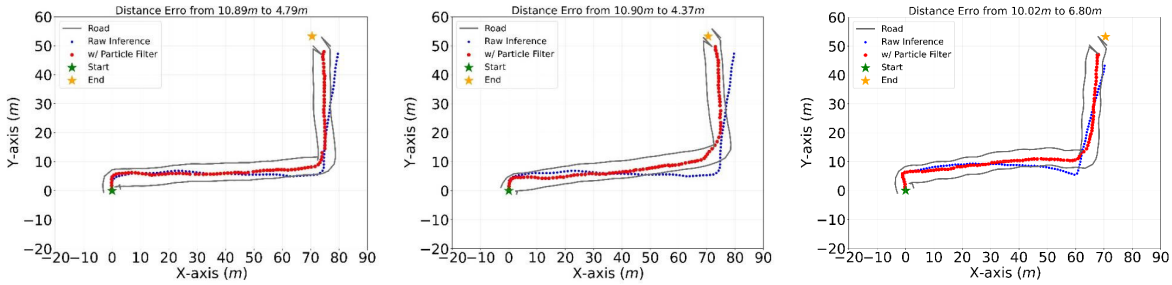


Fig. 21. Example of return navigation result with and without particle filter. With the particle filter the estimated trajectory is within the planned road. (a) Flat. (b) Calling. (c) Bag.

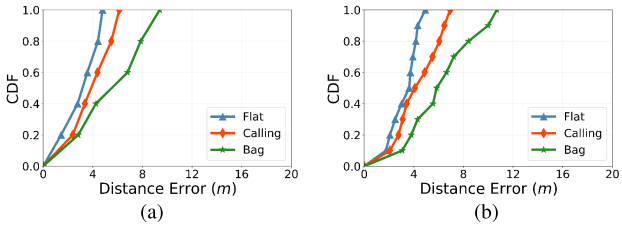


Fig. 22. Result of return navigation error in different postures. The result of flat posture is the best. But even for the worst result, the error is less than 12 m. (a) Campus. (b) Mall.

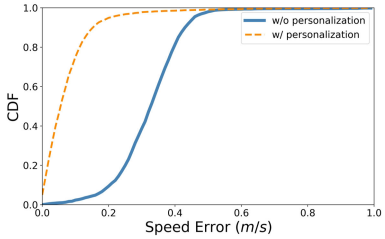


Fig. 23. Comparison between with and without personalized training. After personalized training, the error of speed estimation greatly reduced.

TABLE IV
COMPARISON OF PEDESTRIAN TRACKING RESULT ON DATASET IN SHOPPING MALL AND LONG TRAJECTORY

Dataset	Posture (leaving-return)	Mean	95%	Max
shopping mall	Flat-Flat	3.41	4.66	4.93
	Calling-Flat	5.60	7.49	11.05
	Bag-Flat	7.15	9.73	12.80
long trajectory ($\geq 500m$)	Flat-Flat	19.59	25.54	26.45
	Calling-Flat	28.89	34.78	35.24
	Bag-Flat	39.77	46.65	47.86

on dataset in shopping mall and dataset of long trajectory ($\geq$500 m). From the table, we can see that when the walking distance becomes longer, the error will obviously increase, which indicates that our method is not suitable for long distance walking now.

F. Personalized Training

During the study, we found that the same trained model had different effects when used by different users on different smartphones, which indicated that if we fine-tune the general model for each different user using their own trajectories, the effect would be improved. Therefore, we do the corresponding personalized training experiment.

In this experiment, users were assigned to use the same smartphone for consecutive days to collect outdoor trajectories. At least three trajectories were collected every day by each user, and the collection time of each trajectory was not less than 3 min, and additional data were collected for testing on the final day. The posture of all of smartphones were flat. Fig. 23 shows the evaluation result, and the effect is compared with that of the general model. As figure shows the effect of the personalized model is significantly better than the general model. The mean value of speed error per second for fine-tune is 0.08 m/s, 70% better than the general model and the cdf 95% value of speed error per second for fine-tune is 0.21 m/s, 58% better than the general model. It means that for the general model, it is meaningful to fine-tune the personalized model for each user.

We also performed personalized experiments on the long distance trajectories, and Fig. 24 shows the results of the general model and the fine-tuning model on nine different long distance trajectories. The ordinate of the figure is the value of the difference between the distance errors of the final points

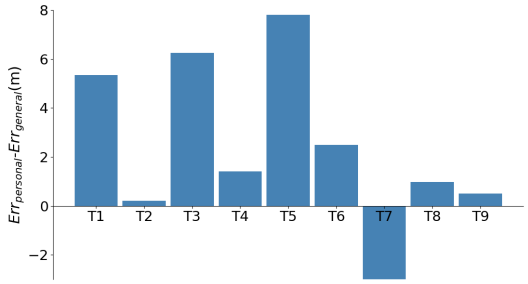


Fig. 24. Comparison between with and without personalized training on long distance trajectories. For nine trajectories, the error of the personalized training model is greater than that of the general model in only one trajectory.

obtained by the two models. $Err_{general}$ is the distance errors of the final points obtained by the general model, and $Err_{personal}$ is the distance errors of the final points obtained by the fine-tune model. Except on the seventh trajectory, the distance error of final point with general model is lower than that of personal training model, and on the remaining eight trajectories, the error of personal model is smaller. Especially at trajectory No. 5, $Err_{personal}$ is lower than $Err_{general}$ nearly 8 m.

In addition, in actual use, we download the general model from the cloud, and then personalized training will be carried out on the user's local device, rather than uploading the user's data to the cloud, which well protects the user's privacy.

What's more, our model is designed to be very lightweight. We use DL4J framework to implement the locally trainable deep model, and do the model compression and pruning. So that the training task will not make too much burden.

VIII. Conclusion

In this article, we propose a navigation algorithm for car finding in underground parking lots. This algorithm is capable of automatically identifying disembarkation behavior, tracking, and recording the user's walking trajectory from the drop-off point, and providing navigation services upon the user's return to the parking lot. Our method relies solely on smartphones, utilizing IMU and GPS data collected outdoors for training purposes. Moreover, it only requires IMU data for inference indoors without any additional equipment or map support.

However, there are several shortcomings in our work. First, our algorithm currently assumes that users enter and exit through the same entrance, which means if users enter and exit through different entrances, it is likely that they will not be able to return to the correct exit position. Second, the natural arm swing posture will cause a significant drift errors of the IMU readings and make it unusable. At present, we have not found a good noise reduction method, so that our method cannot be applied on this posture currently. Third, our proposed method has large error when applied to long-distance trajectories. How to address these challenges will be a focus of our future work.

References

[1] T. Basar, *A New Approach to Linear Filtering and Prediction Problems*. Wiley, 2001, pp. 167–179.

[2] 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.

[3] W. Kang and Y. Han, "SmartPDR: Smartphone-based pedestrian dead reckoning for indoor localization," *IEEE Sensors J.*, vol. 15, no. 5, pp. 2906–2916, May 2015.

[4] N.-H. Ho, P. Truong, and G.-M. Jeong, "Step-detection and adaptive step-length estimation for pedestrian dead-reckoning at various walking speeds using a smartphone," *Sensors*, vol. 16, no. 9, p. 1423, Sep. 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/9/1423

[5] Y. Shu, K. G. Shin, T. He, and J. Chen, "Last-mile navigation using smartphones," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.* New York, NY, USA: Association for Computing Machinery, Sep. 2015, pp. 512–524, doi: 10.1145/2789168.2790099.

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

[7] S. Shen, M. Gowda, and R. Roy Choudhury, "Closing the gaps in inertial motion tracking," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.* New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 429–444, doi: 10.1145/3241539.3241582.

[8] S. Bell, W. R. Jung, and V. Krishnakumar, "WiFi-based enhanced positioning systems: Accuracy through mapping, calibration, and classification," in *Proc. 2nd ACM SIGSPATIAL Int. Workshop Indoor Spatial Awareness*. New York, NY, USA: Association for Computing Machinery, Nov. 2010, pp. 3–9, doi: 10.1145/1865885.1865888.

[9] S. Lee, J. Kim, and N. Moon, "Random forest and WiFi fingerprint-based indoor location recognition system using smart watch," *Human-Centric Comput. Inf. Sci.*, vol. 9, no. 1, p. 6, Feb. 2019, doi: 10.1186/s13673-019-0168-7.

[10] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: Robust IMU double integration," 2017, *arXiv:1712.09004*.

[11] H. Yan, S. Herath, and Y. Furukawa, "RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods," 2019, *arXiv:1905.12853*.

[12] C. Chen, P. Zhao, C. Xiaoxuan Lu, W. Wang, A. Markham, and N. Trigoni, "OxIOD: The dataset for deep inertial odometry," 2018, *arXiv:1809.07491*.

[13] X. Ren et al., "PeTrack: Smartphone-based pedestrian tracking in underground parking lot," in *Proc. 18th Int. Conf. Mobility, Sens. Netw. (MSN)*, Dec. 2022, pp. 752–756.

[14] Z. Li, Y. Shu, B. F. Karlsson, Y. Lin, and T. Moscibroda, "Demo: Towards flexible and scalable indoor navigation," in *Proc. 23rd Annu. Int. Conf. Mobile Comput. Netw.* New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 495–497, doi: 10.1145/3117811.3119854.

[15] Y. Shu, Z. Li, B. Karlsson, Y. Lin, T. Moscibroda, and K. Shin, "Incrementally-deployable indoor navigation with automatic trace generation," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2395–2403.

[16] 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*. New York, NY, USA: Association for Computing Machinery, Jun. 2012, pp. 197–210, doi: 10.1145/2307636.2307655.

[17] Y. Tong et al., "Vehicle inertial tracking via mobile crowdsensing: Experience and enhancement," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–13, 2022.

[18] J. Janardhanan, G. Dutta, and V. Tripuraneni, "Attitude estimation for pedestrian navigation using low cost MEMS accelerometer in mobile applications, and processing methods, apparatus and systems," U.S. Patent 2016/0 290 807 A1, Oct. 6, 2016. [Online]. Available: https://www.freepatentsonline.com/y2016/0290807.html

[19] S. Cortés, A. Solin, E. Rahtu, and J. Kannala, "ADVIO: An authentic dataset for visual-inertial odometry," 2018, *arXiv:1807.09828*.

[20] 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.* New York, NY, USA: Association for Computing Machinery, Sep. 2014, pp. 147–158, doi: 10.1145/2632048.2636079.

[21] M. Abbas, M. Elhamshary, H. Rizk, M. Torki, and M. Youssef, "WiDeep: WiFi-based accurate and robust indoor localization system using deep learning," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom*, Mar. 2019, pp. 1–10.

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

Her research interests include indoor localization, inertial tracking, and mobile crowdsensing.

Shuli Zhu received the B.S. and M.S. degrees in software engineering from Beijing Jiaotong University, Beijing, China, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree in software engineering.

His research interests include mobile computing and vehicle dead reckoning.

Feng Liu received the B.S. degree in software engineering from the Inner Mongolia University of Technology, Hohhot, China, in 2022. He is currently pursuing the M.S. degree in software engineering with Beijing Jiaotong University, Beijing, China.

His research interests include sensor positioning and mobile computing.

Haitao Li received the B.S. degree in software engineering from Beijing Jiaotong University, Beijing, China, in 2023, where he is currently pursuing the M.S. degree in software engineering.

His research interests include smart transportation and machine Learning.

Haohang Li received the B.S. degree in software engineering from Beijing Jiaotong University, Beijing, China, in 2020.

Her research interests include indoor positioning and mobile computing.

Xuan Xiao received the B.S. degree in network engineering from the China University of Mining and Technology, Xuzhou, China, in 2016. He is currently pursuing the Ph.D. degree in software engineering from Beijing Jiaotong University, Beijing, China.

His research interests include mobile computing and applications, and the Internet of Things.

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.

Zhang Zhang received the B.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016, and the master's degree from the Hong Kong University of Science and Technology, Hong Kong, in 2017.

He is currently an Engineer with the China Institute of Electronic Technology Standardization, Beijing, mainly responsible for 2-D code, radio frequency identification (RFID), real-time positioning, unmanned aerial vehicle (UAV) identification, and other related fields of standards and testing research.