2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

ARPDR: An Accurate and Robust Pedestrian Dead Reckoning System for Indoor Localization on Handheld Smartphones

Xiaoqiang Teng[1], Pengfei Xu[1], Deke Guo[2], Yulan Guo[3], Runbo Hu[1], and Hua Chai[1]

***Abstract*— The proliferation of mobile computing has prompted Pedestrian Dead Reckoning (PDR) to be one of the most attractive and promising indoor localization techniques for ubiquitous applications. The existing PDR approaches either suffer position drifts caused by accumulative errors or are sensitive to various users. This paper presents ARPDR, an accurate and robust PDR approach to improve the accuracy and robustness of indoor localization methods. Particularly, we propose a novel step counting algorithm based on motion models by deeply exploiting inertial sensor data. We then combine step counting with adaptive thresholding to personalize the PDR system for different users. Furthermore, we propose a novel stride-heading model with a deep neural network to predict stride lengths and walking orientations, thus the displacement errors are significantly reduced. Extensive experiments on public datasets demonstrate that ARPDR outperforms the state-of-the-art PDR methods.**

I. INTRODUCTION

The past decade has witnessed the conceptualization and development of various indoor localization techniques on handheld smartphones based on Inertial Measurement Unit (IMU) [1]–[3], vision [4]–[7], Wi-Fi signals [8], [9], and magnetic [10]. Among them, IMU-based dead reckoning has been extensively utilized due to its characteristics: low cost, high energy-efficiency, working anywhere, and embedded in every smartphone. As a result, IMU-based dead reckoning has been suggested as one of solutions for ubiquitous indoor localization with a variety of applications, including indoor navigation and augmented reality.

IMU-based approaches achieve positions through the integration of accelerometer and gyroscope readings. The smartphone rotation is obtained from gyroscope readings. The linear acceleration is calculated by subtracting the gravity from accelerometer readings. The rotations and linear accelerations are used to calculate velocities, which are further integrated to produce positions. However, it is challenging for the current integration approaches to work on ordinary smartphones due to *their high error levels of sensors*. The errors of IMU sensors, including noise and bias from the accelerometer and gyroscope, are difficult to be accurately calibrated online or offline. Therefore, they suffer position drifts since errors accumulate rapidly in the integration process [11], [12].

Corresponding author: Deke Guo
[1]Didi Chuxing {tengxiaoqiang, xupengfeipf, hurunbo, chaihua}@didiglobal.com
[2]College of System Engineering, National University of Defense Technology, Changsha, Hunan, P. R. China dekeguo@nudt.edu.cn
[3]College of Electronic Science and Technology, National University of Defense Technology, Changsha, Hunan, P. R. China yulan.guo@nudt.edu.cn

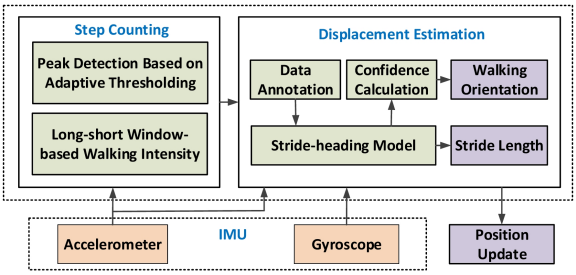


Fig. 1. Main architecture of ARPDR framework.

To alleviate these issues, Pedestrian Dead Reckoning (PDR) and data-driven approaches have been investigated to achieve good performance for position estimation. Recently, many PDR approaches have been developed to estimate positions on handheld smartphones [1], [13], [14]. These approaches are deployed with three stages: step counting [15]–[18], stride estimation [19], [20], and heading estimation [21]–[23]. Despite these extensive efforts, few solutions have been deployed in practical applications due to their high position drifts from cumulative errors, especially for errors of walking orientation [3], [24]. Data-driven approaches estimate positions with nonlinear mapping velocity or position vectors from IMU data using deep neural networks [2], [3], [24]. They provide a new opportunity to improve the performance of inertial navigation.

This paper presents ARPDR, an accurate and robust PDR system for indoor localization using IMU data by combining data-driven and conventional PDR approaches. It is developed on ordinary smartphones to estimate the positions of a user. We first propose a novel step counting algorithm to significantly reduce detection errors of step events and improve the performance of existing PDR approaches. Specifically, we present a peak detection module based on adaptive thresholding which enhances the robustness of the step counting to different users and device positions. We also propose a long-short window-based walking intensity module which improves the accuracy of the step counting. We then propose a novel stride-heading model to predict stride lengths and walking orientations from IMU sensor history simultaneously.

An ARPDR prototype system was implemented on iOS and Android platforms. Extensive experiments conducted on handheld smartphones show that our approach outperforms existing competitors. ARPDR significantly pushes forward the state-of-the-art PDR approaches.

II. RELATED WORK

Our work focuses on sensor-integrated indoor localization. Rotation estimation with IMU-only has been deployed for virtual reality applications, e.g., Google Cardboard VR[1] and Oculus VR[2]. However, the position estimation remains a challenging issue due to errors (e.g., bias and noise) produced by IMU sensors. To reduce position drifts, PDR and data-driven approaches have been extensively investigated by researchers.

Pedestrian Dead Reckoning. The basic idea of the PDR approach is to add the estimated displacement vector to the previously estimated position. To implement a PDR system, three typical steps are employed: step counting, stride estimation, and orientation estimation [1], [13], [14], [25]. The rationality behind step counting is that the accelerations exhibit periodical and repetitive patterns during a user's walking. Various algorithms were proposed to detect step events using accelerometer or gyroscope readings, such as peak detection [15]–[17] and zero-crossing detection [18]. Furthermore, physical distances are calculated by multiplying step counts with stride lengths [13], [19], [20]. To obtain the distance vector, heading estimation is performed using gyroscope and accelerometer readings, which provide the relative direction changes to the phone platform [21]–[23]. Existing PDR approaches suffer position drifts accused by cumulative errors from all of these steps. Therefore, in this paper, cumulative errors are reduced by exploiting novel motion models.

Data-driven dead reckoning. Recently, data-driven approaches have been exploited based on machine learning or deep learning techniques [2], [3], [24]. RIDI was proposed to estimate the positions of a user by regressing the velocity vector in a device coordinate frame [2]. IONet [3] and RoNIN [24] were presented based on deep neural networks to estimate positions by regressing the velocity magnitude. In this paper, ARPDR improves the robustness of position estimation by incorporating PDR and data-driven approaches even for unseen users on datasets.

III. THE PROPOSED APPROACH

ARPDR consists of two modules: step counting and displacement estimation. The step counting is to detect a user's step events. The displacement estimation module is to calculate the traveled distances from stride lengths and walking orientations within a step. All modules exploit accelerometer and gyroscope readings embedded in a smartphone. Figure 1 shows the main architecture of ARPDR.

A. Step counting algorithm

The proposed algorithm is based on peak detection [15] due to its good performance under different usage positions of a smartphone, e.g., holding in hand or bag. Each walking step of a user corresponds to a peak value of the accelerometer readings. Therefore, traveled steps are calculated by detecting and counting peak values. However, the peak detection algorithm suffers considerable false positive and false negative errors which will be significantly accumulated [15], [16]. In this paper, the performance of the peak detection algorithm is further improved by applying two modules, including the peak detection module based on adaptive thresholding and the long-short window-based walking intensity module.

Peak detection based on adaptive thresholding. A peak value is a local maxima, which is extracted from amplitudes of the acceleration. A candidate step event is then detected by checking whether the peak value exceeds a predefined threshold [15], [16]:

$$S_t^{peak} = \begin{cases} 1, & \text{if } a_t \geq TH_{peak}, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where, a_t is the amplitude of acceleration, i.e., $a_t = \sqrt{a_{t,x}^2 + a_{t,y}^2 + a_{t,z}^2}$. $a_{t,x}$, $a_{t,y}$, and $a_{t,z}$ are the acceleration components along the x, y, and z axes in the IMU's body frame at time t, respectively. TH_{peak} is the threshold of peak detection. Existing algorithms fix the threshold, thus it offers detection errors for a variety of users. In ARPDR, TH_{peak} is a dynamically adjusted threshold. Given an initial value (set to 1.2 m/s^2), the peak threshold is calculated as:

$$TH_{peak} = \max\{1.2, \frac{a_{t-2} + \cdots + a_t + \cdots + a_{t+2}}{4}\}, \tag{2}$$

where, a_t is the amplitude of acceleration at the time t.

Long-short window-based walking intensity. Walking intensity is used to measure the variance of the acceleration within one step. It is essential for detecting a step event because the intensity presents a probability distribution of step events due to a periodical and repetitive pattern of accelerations. If the walking intensity exceeds a predefined threshold, the acceleration on this moment is considered as a candidate step event:

$$S_t^{walking} = \begin{cases} 1, & \text{if } \sum_{t=0}^{n_w} W_t \geq TH_{walking} \text{ and } a_{avg}^{short} > a_{avg}^{long}, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where $W_t = (a_{avg}^{short} - a_{avg}^{long})^2 \Delta t$, Δt is the sampling interval of the acceleration, n_w is the sampling number of the acceleration starting with the last step, and $TH_{walking}$ is a threshold (empirically set to 0.3 in our work). a_{avg}^{short} and a_{avg}^{long} are the average amplitude of the acceleration on short and long time window. The size of short and long time windows are empirically set to 0.2 second and 1.0 second, respectively.

As shown in Fig. 2, a red star represents a step point and a blue dot represents the value of the walking intensity. It demonstrates that the proposed method detects the step event successfully. Once a step event has been detected, values of the walking intensity will be reset to 0.

In summary, the resulting step event S_t^{step} is detected from the intersection of the above conditions:

$$S_t^{step} = S_t^{peak} \cap S_t^{walking}. \tag{4}$$

[1]https://arvr.google.com/cardboard/

[2]https://www.oculus.com/

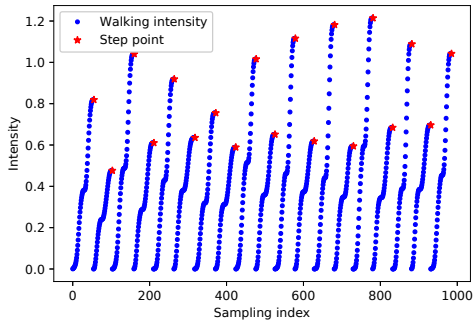


Fig. 2. An illustration of the walking intensity. A red star represents a step point and a blue dot represents the value of the walking intensity on each frame. Once a step event has been detected, values of the walking intensity will be reset to 0.

B. Displacement estimation

The step events are typically converted into displacements by multiplying user's stride lengths and walking orientations. Orientation estimation is more challenging since walking orientations are different from device orientations. Device orientations are derived by the Complementary Filter (CF)-based [22], [23] or the EKF-based [26] algorithms using accelerometer and gyroscope readings. It depends on the held position of a smartphone, such as landscape or portrait. For example, the deviation between the walking orientation and the device orientation maybe 90° when a user walks by holding a smartphone on one side. For stride estimation, existing methods have been proposed based on the walking frequency and acceleration variance in a time window [13], [19]. They produce stride estimation for a variety of users with a mean error of 6.0cm. Those orientation deviations and stride length errors cause position drifts for PDR approaches.

To reduce these errors, a stride-heading model is proposed based on Temporal Convolutional Network (TCN) [27] architecture in ARPDR. It predicts stride lengths and walking orientations using accelerometer and gyroscope readings with two key design principles: (i) Ground-truth annotation, (ii) Velocity and angle observations. We now explain these principles.

Ground-truth annotation. The ground-truth values of stride lengths and walking orientations are calculated with walking traces derived from a 3D tracking sensor, e.g., a smartphone with Apple ARKit[3] or Google ARCore[4]. Given a time window size s (e.g., 50 for 100Hz of sampling frequency), the walking orientation θ_t at the t-th frame is calculated as:

$$\theta_t = \arctan \frac{p_{t+s,x} - p_{t,x}}{p_{t+s,y} - p_{t,y}}, \tag{5}$$

where $p_{t,x}$ and $p_{t,y}$ are coordinates along the x and y axes, respectively. An illustration of the walking orientation is shown in Fig. 3.

[3]https://developer.apple.com/augmented-reality/
[4]https://developers.google.com/ar/

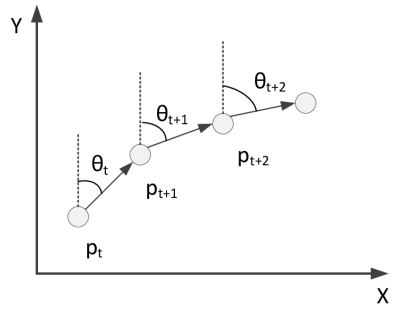


Fig. 3. An illustration of the walking orientation. A circle represents a step point.

The stride length L_t at the t-th frame is calculated as:

$$L_t = \sqrt{(p_{t+s,x} - p_{t,x})^2 + (p_{t+s,y} - p_{t,y})^2}, \tag{6}$$

Stride-heading model. We use TCN [28] as a backbone to predict a 7D vector $(v_x, v_y, \sin\theta, \cos\theta, \sin\phi, \cos\phi, l_{x,y})$ with IMU data from a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis linear accelerometer. We assume that users walk on a horizontal plane, i.e., the $x-y$ plane. A 7D vector consists of a 2D velocity vector (v_x, v_y), a 1D stride length vector $l_{x,y}$, a 2D vector of the walking orientation $(\sin\theta, \cos\theta)$, and a 2D vector of the orientation deviation $(\sin\phi, \cos\phi)$. θ and ϕ are the angles of the walking orientation and the orientation deviation, respectively. The orientation deviation is an angle between the device orientation and the walking orientation. Therefore, at frame i, the network takes IMU data from frame $i-W$ to i as a $W*9$ tensor and produces 7D vector at frame i, where W is the time window depending on the sampling frequency of IMU sensors. At the beginning part of TCN, a bilinear layer, a ReLU activation function, and a dropout are added to extract features from IMU data. At the end part of the TCN, a fully connected layer is attached. Details of the proposed stride-heading model are shown in Fig. 4.

For each input sample $\boldsymbol{x}$, the predicted output of the network is $\boldsymbol{o} = (v_x, v_y, \sin\theta, \cos\theta, \sin\phi, \cos\phi, l_{x,y})$. Loss L_o is calculated as follows:

$$L_o = \frac{1}{2\sigma_1^2} L_p + \frac{1}{2\sigma_2^2} L_v + \frac{1}{2\sigma_3^2} L_h + \frac{1}{2\sigma_4^2} L_s + \log \sigma_1 \sigma_2 \sigma_3 \sigma_4, \tag{7}$$

where L_p, L_v, L_h, and L_s are losses of the walking path, the velocity, the orientation, and the stride length, respectively. σ_1, σ_2, σ_3, and σ_4 are weights of the loss. These weights are learnable parameters [29].

Losses L_p, L_v, L_h, and L_s are calculated by Mean Square Error (MSE) as follows:

$$L_p = \frac{1}{2N} \sum_{i=1}^{N} (p_i - \bar{p}_i)^2, \tag{8}$$

$$L_v = \frac{1}{2N} \sum_{i=1}^{N} (v_i - \bar{v}_i)^2, \tag{9}$$

$$L_h = \frac{1}{2N} \sum_{i=1}^{N} [(h_i - \bar{h}_i)^2 + (d_i - \hat{d}_i)^2], \tag{10}$$

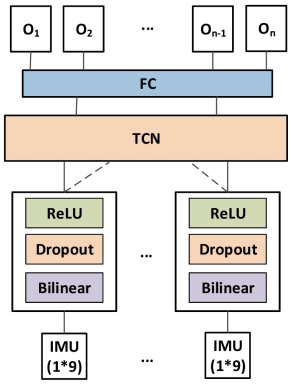


Fig. 4. The network architecture for the stride-heading model.

$$L_s = \frac{1}{2N}\sum_{i=1}^{N}(l_i - \bar{l}_i)^2, \tag{11}$$

where $\bar{p}_i$ and p_i are the estimated and ground-truth positions in a path by integrating velocities, respectively. $\bar{v}_i$, $\bar{h}_i$, $\bar{d}_i$, and $\bar{l}_i$ are the estimated velocity, walking orientation, orientation deviation, and stride length, respectively. v_i, h_i, d_i, and l_i are the ground-truth velocity, walking orientation, orientation deviation, and stride length, respectively.

Velocity and angle observations. A confidence S_i^{conf} at the i-th frame is calculated by the predicted velocity magnitude, walking orientation θ_i, and orientation deviation ϕ_i as follows:

$$\begin{aligned} S_i^{conf} = \frac{1}{||v_i||}((\sin\theta_i - \sin(\alpha_i - \phi_i))^2 \\ +(\cos\theta_i - \cos(\alpha_i - \phi_i))^2), \end{aligned} \tag{12}$$

where α_i is the device orientation at the i-th frame calculated by the CF-based algorithm [22], [23]. $||v_i||$ is the velocity magnitude. If S_i^{conf} is less than a predefined threshold, the predicted walking orientation is considered as the final walking orientation θ_i. Otherwise, the device orientation α_i is considered as the final walking orientation.

C. Put it all together

We incorporate the results from the step counting and the displacement estimation to continuously estimate the positions of a user. Thus, the current position P_i at the i-th step is calculated by adding the displacement vector to the previous position P_{i-1}:

$$P_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} + L_i \begin{bmatrix} \sin\theta_i \\ \cos\theta_i \end{bmatrix}, \tag{13}$$

where x_i and y_i represent coordinates at the i-th step along the x and y axes in the global coordinate system. L_i and θ_i are the stride length and the walking orientation at the i-th step.

IV. EXPERIMENTAL EVALUATION

In this section, we first evaluated the performance of step counting, stride estimation, and orientation estimation, and then tested the performance of the position estimation.

A. Experimental setting

We evaluated the performance of proposed algorithms on three datasets, including two public datasets and a new stride length dataset, as described follows.

Oxford Step Counter dataset. The Oxford Step Counter dataset was collected from three researchers walking with three different phones (Google Pixel, Samsung Galaxy S6, LG Nexus 5) held in different positions: in a hand, a pocket, an armband, a shoulder purse, and a neck pouch on a lanyard [15], [17]. It contains about 60 trajectories.

RoNIN. RoNIN published 140 trajectories acquired by 100 users [24]. The ground-truth positions were obtained using a 3D tracking device. The IMU data was collected using three smartphones (Asus Zenfone AR, Samsung Galaxy S9, and Google Pixel 2 XL) in 3 buildings with motion free. Note that, they only published 50% of the dataset[5].

Stride Length dataset. Due to the lack of a public stride length dataset, we built a new stride length dataset, which contains 100 trajectories from 11 users with iPhone 8P, iPhone 7P, Samsung Galaxy S7, and Huawei Mate9 smartphones held in hand. The ground-truth stride lengths are recorded by a 3D tracking device [2], [24].

ARPDR system was implemented in C++ for iOS and Android platforms and tested on different smartphones. The hyper-parameters of step counting, stride estimation, and orientation estimation algorithms were trained on the training dataset. Our stride-heading model was implemented with PyTorch[6]. The TCN network has six residual blocks with 32, 64, 128, 256, 72, and 36 channels and a convolutional kernel of size 3. The TCN is followed by a fully connected layer that output the scalars of velocity, walking orientation, and orientation deviation. For training, a batch size of 256, an initial learning rate of 0.0003, ADAM optimizer, and dropout strategy with probability 0.2 were used.

B. Performance evaluation for key components

Performance of step counting. We conducted quantitative evaluations of the proposed step counting algorithm on the Oxford Step Counter dataset with the competing method [16]. The S_{Acc} metric is used to measure the performance of the proposed step counting algorithm:

$$S_{Acc} = (1 - \frac{\min\{S_T, |S_E - S_T|\}}{S_T}) \times 100\%, \tag{14}$$

where S_E and S_T are the estimated and ground-truth number of walking steps, respectively.

It can be seen from Table I, our ARPDR outperforms the baseline [16], achieving about 2.3% accuracy improvements. It demonstrates that peak detection based on adaptive thresholding and long-short window-based walking intensity modules significantly improve the accuracy of the step counting. As shown in Fig. 5, red stars represent step points and blue dots represent thresholds. We can see that thresholds are dynamically adjusted with the user's walking.

[5]https://ronin.cs.sfu.ca/README.txt
[6]https://pytorch.org/

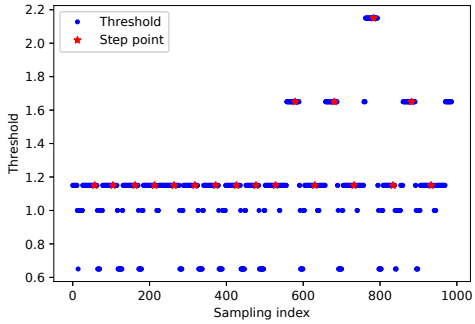


Fig. 5. An illustration of the adaptive thresholding strategy. Red stars represent step points and blue dots represent thresholds detected from the step counting algorithm.

Performance of stride estimation. The performance of the stride estimation from the stride-heading model was tested on the Stride Length dataset with three published methods, including Weinberg [19], Zhang et al. [13], and ANN [20]. Table II summarizes errors of the stride length for ARPDR and the other methods. It can be seen that ARPDR presents lower errors than other methods with big margins. ARPDR achieves average stride length errors of 2.2cm. Other methods achieve average stride length errors of 10.0cm, 8.1cm, and 6.1cm, respectively. These results confirm that the TCN network is highly learning capable of stride lengths.

Performance of orientation estimation. Furthermore, we conducted quantitative evaluations of the proposed stride-heading model on the RoNIN dataset with two published methods: Madgwick et al. [22] and RoNIN [24]. Madgwick et al. [22] estimated device orientations using the Complementary Filter algorithm. RoNIN [24] took the LSTM architecture to predict a 2D vector (x,y), which are sin and cos values of the body heading angle at each frame. To test the performance of walking orientations on aside and backward traces, we transformed the IMU data with a random angle (e.g., 90° or 150°) on some datasets [14].

ARPDR achieves average orientation error of 18.21°, while Madgwick et al. [22] and RoNIN obtain average orientation errors of 65° and 19.88°, respectively. ARPDR achieves the best performance. That is because that the walking orientation during users' walking is smooth and the confidence can filter outliers of predicted orientations from the stride-heading model.

C. Position evaluation

The Absolute Trajectory Error (ATE) is used to evaluate the performance of the position estimation. ATE is defined as the Root Mean Squared Error (RMSE) between the estimated and ground-truth trajectories as a whole. We compared our method with three methods:

(i) **PDR** consists of the step counting [16], the stride estimation [19], and the device orientation [22] algorithms.

(ii) **SH-PDR** consists of the proposed step counting in ARPDR, the stride estimation [19], and the body heading [24] algorithms.

(iii) **RoNIN**: We use the official implementation [24]. Since the work only published 50% of the dataset, we split the dataset into train, validation, and test datasets by $6:1:3$. We trained a model in the training dataset.

Table III lists results of the position estimation. The RoNIN dataset provides two testing sets, one for subjects that are also included in the training set and the other for unseen subjects. It can be seen that ARPDR outperforms competing approaches with significant margins, achieving about 20% accuracy improvements over the state-of-the-art on the RoNIN dataset. We can notice that our ARPDR generalizes well from experimental results on unseen testing set, in particular, ATE and RTE below 5.0 and 3.6, respectively. Our future work is to push the limit of generalization capability by designing a better regression network.

Figure 6 shows the trajectory achieved by our ARPDR is well aligned over one obtained by ARKit with an iPhone 8P in real scenes. It demonstrates that ARPDR achieves a good performance of position estimation.

TABLE I
ACCURACY OF THE STEP COUNTING ALGORITHM

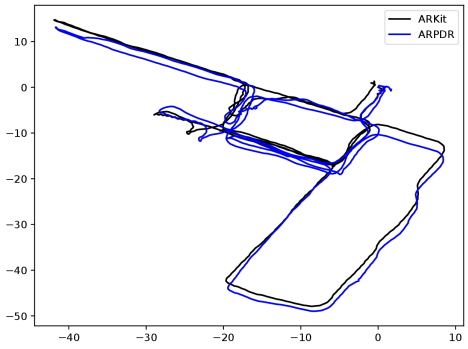
Dataset	Thanh et al. [16]	ARPDR
User 1, hand	99.08%	100%
User 1, front pocket	99.39%	99.69%
User 1, back pocket	95.34%	99.71%
User 1, neck pouch	97.98%	99.71%
User 1, bag	96.24%	99.71%
User 1, armband	100%	100%
User 2, hand	97.65%	99.41%
User 2, front pocket	95.92%	98.83%
User 2, back pocket	99.41%	98.81%
User 2, neck pouch	96.67%	99.72%
User 2, bag	89.2%	98.89%
User 2, armband	97.67%	98.54%
Average Accuracy	97.04%	99.42%

TABLE II
STRIDE LENGTH ERRORS

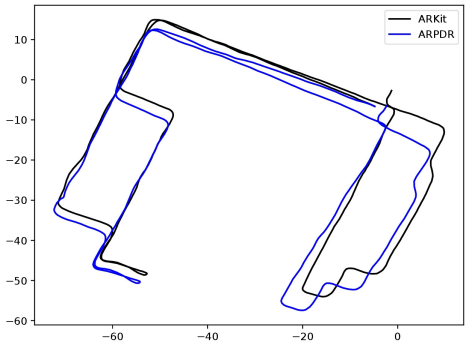
Algorithm	Average Error
Weinberg [19]	10.0cm
ANN [20]	8.1cm
Zhang et al. [13]	6.1cm
ARPDR	2.2cm

TABLE III
POSITION EVALUATION. EACH ENTRY SHOWS THE MEAN POSITIONAL ERROR

Test Subjects	Metric	PDR	SH-PDR	RoNIN [24]	ARPDR
Seen	ATE	28.10	10.07	5.78	4.10
	RTE	20.60	9.92	3.68	3.52
Unseen	ATE	26.17	13.99	6.73	5.00
	RTE	20.70	12.08	4.33	3.60



(a) Length: 358m, ATE: 1.2



(b) Length: 397m, ATE: 2.6

Fig. 6. Evaluation in the real scene. The black line marks the trajectory given by the ARKit system using an iPhone 8P smartphone and the blue line marks the trajectory given by our system.

V. CONCLUSIONS

In this work, we propose ARPDR, an accurate and robust PDR system for inertial navigation to estimate positions of smartphones by exploiting mobile computing and deep learning techniques. We have derived robust step counting and displacement estimation algorithms by deeply exploiting human motion patterns. ARPDR is fed with IMU data only and requires no other sensor. The proposed method has shown significant performance improvement in public datasets and real scenes.

ACKNOWLEDGMENT

This work is partially supported by the National Key Research and Development Program of China under Grant No. 2018YFE0207600 and the National Natural Science Foundation of China (No. 61972435).

REFERENCES

[1] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proc. of ACM MobiSys*, 2012.

[2] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: robust IMU double integration," in *Proc. of Springer ECCV*, 2018.

[3] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONet: Learning to cure the curse of drift in inertial odometry," in *Proc. of AAAI*, 2018.

[4] J. Dong, Y. Xiao, M. Noreikis, Z. Ou, and A. Ylä-Jääski, "iMoon: Using smartphones for image-based indoor navigation," in *Proc. of ACM SenSys*, 2015.

[5] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[6] X. Teng, D. Guo, Y. Guo, X. Zhou, and Z. Liu, "CloudNavi: Toward ubiquitous indoor navigation service with 3d point clouds," *ACM Transactions on Sensor Networks*, vol. 15, no. 1, pp. 1:1–1:28, 2019.

[7] X. Teng, D. Guo, Y. Guo, X. Zhao, and Z. Liu, "SISE: self-updating of indoor semantic floorplans for general entities," *IEEE Transactions on Mobile Computing*, vol. 17, no. 11, pp. 2646–2659, 2018.

[8] M. Kotaru, K. R. Joshi, D. Bharadia, and S. Katti, "SpotFi: Decimeter level localization using WiFi," in *Proc. of ACM SIGCOMM*, 2015.

[9] S. Kumar, S. Gil, D. Katabi, and D. Rus, "Accurate indoor localization with zero start-up cost," in *Proc. of ACM MobiCom*, 2014.

[10] Y. Shu, K. G. Shin, T. He, and J. Chen, "Last-Mile navigation using smartphones," in *Proc. of ACM MobiCom*, 2015.

[11] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using allan variance," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 1, pp. 140–149, 2008.

[12] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold preintegration for real-time visual-inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[13] H. Zhang, W. Yuan, Q. Shen, T. Li, and H. Chang, "A handheld inertial pedestrian navigation system with accurate step modes and device poses recognition," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1421–1429, 2015.

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

[15] B. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proc. of UbiComp*, 2013.

[16] P. Van Thanh, N. Duc Anh, D. Nhu Dinh, P. Hong Hai, T. Van An, S. umbesan, and T. Duc-Tan, "Highly accurate step counting at various walking states using low-cost inertial measurement unit support indoor positioning system," *Sensors*, vol. 18, no. 10, p. 3186, 2018.

[17] D. Salvi, C. Velardo, J. Brynes, and L. Tarassenko, "An optimised algorithm for accurate steps counting from smart-phone accelerometry," in *Proc. of the 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2018.

[18] R. de Silva, J. Perera, C. P. Abeysingha, and N. Abhayasinghe, "A gyroscopic data based pedometer algorithm with adaptive orientation," in *Proc. of the 14th IEEE International Conference on Control and Automation*, 2018.

[19] H. Weinberg, "Using the ADXL202 in pedometer and personal navigation applications," 2002.

[20] H. Xing, J. Li, B. Hou, Y. Zhang, and M. Guo, "Pedestrian stride length estimation from IMU measurements and ANN based algorithm," *Journal of Sensors*, vol. 2017, pp. 6 091 261:1–6 091 261:10, 2017.

[21] P. Zhou, M. Li, and G. Shen, "Use it free: Instantly knowing your phone attitude," in *Proc. of ACM MobiCom*, 2014.

[22] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *Proc. of IEEE International Conference on Rehabilitation Robotics*, 2011.

[23] M. Euston, P. W. Coote, R. E. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," in *Proc. of IEEE/RSJ IROS*, 2008.

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

[25] X. Teng, D. Guo, Y. Guo, X. Zhou, Z. Ding, and Z. Liu, "IONavi: An indoor-outdoor navigation service via mobile crowdsensing," *ACM Transactions on Sensor Networks*, vol. 13, no. 2, pp. 12:1–12:28, 2017.

[26] J. L. Marins, X. Yun, E. R. Bachmann, R. B. McGhee, and M. Zyda, "An extended kalman filter for quaternion-based orientation estimation using MARG sensors," in *Proc. of IEEE/RSJ IROS*, 2001.

[27] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. of IEEE CVPR*, 2018.