# Particle Filter Reinforcement via Context-Sensing for Smartphone-based Pedestrian Dead Reckoning

Wenhua Shao, Fang Zhao, Haiyong Luo, Hui Tian, Jiaxin Li, Antonino Crivello

***Abstract*— Pedestrian dead reckoning based on particle filter is commonly used for enabling seamless smartphone-based indoor positioning. However, compass directions indoor are heavily distorted due to the presence of ferromagnetic materials. Conventional particle filters convert the raw compass direction to a distribution adding a constant variance noise and leveraging a particle swarm to simulate the distribution. Finally, the selection of eligible directions is performed applying external constraints mainly imposed from the indoor map. However, the choice of a constant parameter decreases the positioning performances because the variance of nearby context, including topography, ferromagnetic materials, and particle distribution, is not represented. Therefore, we propose the particle filter reinforcement able to adaptively learn and adjust the variance of the direction observing the context in real-time. Experiments in real-world scenarios show that the proposed method improves the positioning accuracy by more than 20% at the 80% probability compared with state-of-the-art methods.**

***Index Terms*— Indoor location tracking, particle filter, pedestrian dead reckoning, reinforcement learning, smartphone-based navigation.***

## I. Introduction

POSITIONING techniques based on inertial measurement unit (IMU) have become widely adopted in the indoor positioning field through the development of smartphones with the micro-electro-mechanical system (MEMS) integrated. Considering the low accuracy of the MEMS in smartphones, traditional inertial navigation systems (INS) are generally not usable [1]. Several researchers have shown pedestrian dead reckoning (PDR) systems able to leverage step detection [2], the prior knowledge of human step length [3] and moving direction to estimate positions and to achieve good accuracies. Nevertheless, the estimation of the moving direction is still challenging in indoor scenarios due to the geomagnetic field heavily distorted by the ferromagnetic materials present in the environment. Therefore, researchers began to consider a direction distribution, leveraging a particle swarm to represent the possible moving directions and finally selecting the correct direction through the map constraints. the uniformity of floor plans.

Estimating an optimal direction variance, able to properly cover all the possible directions, is still challenging. In fact, a small variance simplifies the particle distribution but tends to miss correct directions. A large variance improves the system robustness but decreases the positioning accuracy because the particle is sparse. Several factors affect the optimal variance estimation, including topography, ferromagnetic materials, and particle distribution. In order to simplify this multi-dimensional and multi-channel problem and to find a solution, researchers generally select a constant and empirical variance value based on their experiments [4, 5]. This choice leads to a loss in terms of accuracy and robustness

We address this problem by proposing a novel particle filter reinforcement (PFR). Considering the success of AlphaGo Zero [6], reinforcement learning is able to improve itself through self-play iterations. Therefore, the proposed system observes the state of the particle distribution and the local floor map, then learns a proper action to adjust the variance of particle moving directions exploiting multiple times particles self-moving. The learning process is guided by a reward that encourages at each step the particles to distribute around ground-truth positions. Basically, actions that assure longer tracking time and better positioning accuracy are rewarded. As a conclusion, the contributions of this paper are as follows.

- Particle filter reinforcement via context-sensing. Our algorithm enables the particle filter to automatically adjust the particle distribution by observing the current system contexts. This improves positioning accuracy and robustness.
- Neural network for implementing a particle parameter adjusting policy (PPAP). We propose a lightweight neural network for implementing the PPAP agent in order to quickly converge and effectively evaluate proper actions.
- We have conducted real-world experiments to test our algorithm. Experiments reveal that our algorithm improves the accuracy by more than 20% if compared to traditional particle filters.

The rest of the paper is organized as follows. Section II provides an overview and explain the workflow of the PFR system. Section III details the learning process and the proposed PPAP neural network. Section IV and V present the experiments and conclusions.

This work was supported in part by the National Key Research and Development Program under Grant 2019YFC1511400, the Action Plan Project of the Beijing University of Posts and Telecommunications supported by the Fundamental Research Funds for the Central Universities under Grant 2019XD-A06, the National Natural Science Foundation of China under Grant 61872046, the Joint Research Fund for Beijing Natural Science Foundation and Haidian Original Innovation under Grant L192004, Beijing Natural Science Foundation under Grant 4212024, the Key Research and Development Project from Hebei Province under Grant 19210404D, the Science and Technology Plan Project of Inner Mongolia Autonomous Regio under Grant 2019GG328 and the Open Project of the Beijing Key Laboratory of Mobile Computing and Pervasive Device.

Wenhua Shao, Fang Zhao, and Jiaxin Li are with the School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing, 100876, China. (e-mail: shaowenhua@ict.ac.cn, zfsse@bupt.edu.cn, jiaxin_li@bupt.edu.cn).

Haiyong Luo is with the Institute of Computing Technology, and Beijing Key Laboratory of Mobile Computing and Pervasive Device, Chinese Academy of Sciences, Beijing, 100876, China (e-mail: yhluo@ict.ac.cn).

Wenhua Shao, and Hui Tian are with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, 100876, China. (e-mail: tianhui@bupt.edu.cn).

Antonino Crivello is with the Institute of Information Science and Technologies, CNR, Italy. (e-mail: antonino.crivello@isti.cnr.it).

## II. System Overview

The proposed system architecture, shown in Fig.1, relies on three functional modules: (a) PPAP neural network, (b) PPAP reinforcement learning module and (c) particle filter. The system is driven by pedometer events [2]. When a step event is detected, the estimated step length [3], the moving direction, and the reachable areas at the floor plan considered are given as input. Then, the system estimates user positions. The three functional modules are detailed as follows.

*(a) Particle filter:* this module leverages discrete random particles to approximate the probability of a user's position distribution [4]. Particles move dynamically based on the estimated step length and the measured moving direction. As a consequence, an accurate and standard step length estimation method has been developed [3]. However, the moving direction is difficult to be precisely estimated for smartphone-based systems because the magnetic field inside a building is heavily distorted by ferromagnetic materials (e.g., pillars and large iron cabinets). The distortion degree also varies at different places and, in this paper, we propose the PPAP neural network to dynamically estimate the direction divergence based on the distribution of the particles.

*(b) PPAP neural network:* based on the distribution of particles and the local map of reachable areas, this module estimates the particle direction divergence to optimize the tracking process of the particle filter.

*(c) PPAP reinforcement learning module:* during the training phase (dash lines in Fig.1), the module evaluates the estimation based on the particle distribution and the ground truth position of training samples instructing the PPAP neural network to learn how to manipulate the direction divergence of the particles.

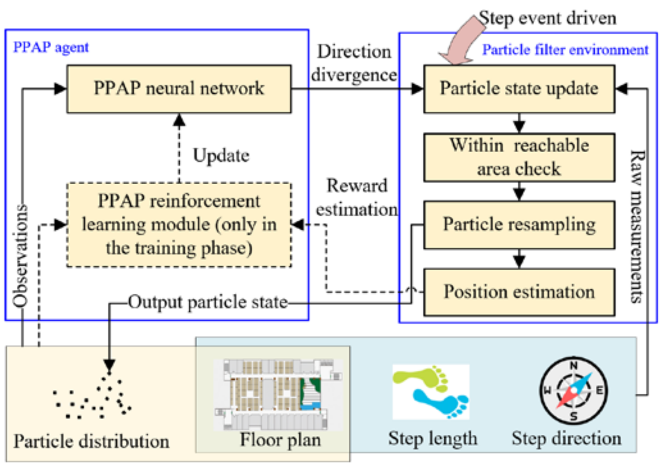


Fig. 1. Overview of the proposed particle filter reinforcement learning.

The workflow of the proposed positioning system based on particle filter reinforcement learning is summarized in Algorithm I.

Algorithm I
The positioning workflow of the proposed system

```
INPUT: Estimated step length $l$. Measured moving direction $\alpha$. Particle number $K$, Local map of reachable areas.
OUTPUT: Estimated user position $(\hat{x}, \hat{y})$.
 1: Initialization
 2: Spread $N$ particles at the initial position $p_0 = (x_0^t, y_0^t)$.
 3: Particle filter reinforcement loop
 4: while True do
 5:   if the $t^{th}$ step event is detected then
 6:     Get a raw measurement of moving direction $\alpha^t$ and step length $l^t$.
 7:     The PPAP neural network estimates particle moving direction divergence $a_t$.
 8:     for each particle $i$ $(1 \le i \le K)$ do
 9:       Get a sample of moving direction $\alpha_i^t$. $\alpha_i^t \sim N(\alpha^t, a_t)$.
10:       Move the particle $(x_i^t, y_i^t) = (x_i^{t-1}, y_i^{t-1}) + l^t(\cos(\alpha_i^t), sin(\alpha_i^t))$.
11:       Set particle state to dead if it moves into unreachable areas.
12:     end for
13:     Resample particles based on alive ones with systematic method [7].
14:     Send the particle distribution to the PPAP neural network.
15:     Output positioning estimation. $(\widehat{x^t}, \widehat{y^t}) = \frac{1}{N}\sum_N (x_i^t, y_i^t)$.
16:   end if
17: end while
```

## III. Reinforcement Learning for the Manipulation of Particle Filters

Through the observation of particle filter state, the PPAP neural network estimates the direction divergence and produces a proper action as input for the particle filter update [8]. The state $\mathcal{S}_t$ is a representation of the current particles' distribution at step $t$, as shown in equation (1). This state is a random variable of 3D tensor which includes the particle distribution matrix $M_P$ and the map of the local reachable area $M_R$. Elements of $M_P$ are integers to indicate the number of particles at a given point. The elements of $M_R$ consists of 1 and 0 indicating wheatear a location is available for walking or not.

$$\mathcal{S}_t = \{M_P, M_R\}, t \in \mathbb{N}^+ \qquad (1)$$

The action $\mathcal{A}_t$ is the decision taken by the PPAP neural network to adjust the movement of particles at the next step. As equation (2) reveals, $\mathcal{A}_t$ is a discrete random variable that indicates the measurements of the particle divergence. The constant value $\sigma_i (1 \le i \le K)$ indicates the standard deviation of the moving direction for each particle.

$$\mathcal{A}_t \in \{\sigma_1, \sigma_2, \dots, \sigma_i, \dots, \sigma_K\}, t \in \mathbb{N}^+ \qquad (2)$$

The PPAP neural network is the key intelligent module, able to learn how to control the particle filter. The learning process is guided by the PPAP reinforcement learning module.

During the training phase, given a series of samples with measured compass directions, estimated step lengths, and known locations, particles move towards the destination. At each step, based on the particle state $\mathcal{S}_t$ and the action $\mathcal{A}_t$, the PPAP reinforcement learning module calculates an environment reward $R_t$. As equation (3) reveals, $R_t$ is proportional to the ratio of particles around the ground truth. Variable $\varepsilon$ defines the threshold for what is considered a close distance. $numel$ is a function that counts the number of eligible elements. Considering that particle filter performances drop significantly if no particle is around the ground truth position, the system prefers actions that create more particles close to the ground truth.

$$R_t(\mathcal{S}_t, \mathcal{A}_t) = \frac{\underset{1\le i\le N}{numel}\left(\sqrt{(x_i^t - x_0^t)^2 + (y_i^t - y_0^t)^2} < \varepsilon\right)}{N}, t \in \mathbb{N}^+ \qquad (3)$$

The system optimization goals are making accurate estimation but also keeping the target tracked as long as possible. As shown in equation (4), we penalize actions that lost tracking by setting their returns to -1. $\gamma^{i-t}$ is the reduction factor of future rewards. $M$ is the total number of steps contained in the training record. Basically, the PPAP reinforcement learning module leverages the reduced return $U_t$ as the loss of the PPAP neural network,

$$U_t(\mathcal{S}_t,\dots,\mathcal{S}_M,\mathcal{A}_t,\dots,\mathcal{A}_M)=\begin{cases}-1 & (all\ particles\ dead)\\ \sum_{i=t}^{M}\gamma^{i-t}R_i(\mathcal{S}_i,\mathcal{A}_i) & (otherwise)\end{cases} \quad (4)$$

As equation (5) reveals, the PPAP neural network can be modelled as a probability of an action instance $a_t$, given a state instance $s_t$ and parameters $\theta$, also called policy, of the neural network.

$$\pi(a_t|s_t;\theta)\triangleq P(\mathcal{A}_t=a_t|\mathcal{S}_t=s_t), t\in\mathbb{N}^+ \quad (5)$$

Consequently, the action value of a state and the action instance pair $(s_t,a_t)$ can be drawn as equation (6). $Q_\pi$ is the sum of the current reward and the expectation of future returns.

$$Q_\pi(s_t,a_t)=R_t(s_t,a_t)+\mathbb{E}(U_{t+1}) \quad (6)$$

Calculating the expectation with respect to different actions, the state value can be drawn in equation (7). $V_\pi$ is the average return of different actions given a state $s_t$ and policy $\theta$.

$$V_\pi(s_t)=\sum_{a_t\in\mathcal{A}_t}\pi(a_t|s_t;\theta)\cdot Q_\pi(s_t,a_t) \quad (7)$$

Eliminating the $s_t$ variable by calculating the expectation of different states, we evaluate the object function in equation (8). In other words, the learning object function maximizes the expectation of the system state value by adjusting the value of policy $\theta$.

$$\begin{aligned}J(\theta)&=\max_\theta \mathbb{E}_{\mathcal{S}_t}(V_\pi(\mathcal{S}_t))\\ &=\max_\theta\sum_{s_t\in\mathcal{S}_t,a_t\in\mathcal{A}_t}P(\mathcal{S}_t=s_t)\cdot\pi(a_t|s_t;\theta)\cdot Q_\pi(s_t,a_t)\end{aligned} \quad (8)$$

In order to maximize the object function $J(\theta)$, we can update $\theta$ along its gradients with the policy gradient method [9]. $\eta$ is the learning rate.

$$\theta_{new}\leftarrow\theta+\eta\cdot\nabla_\theta J(\theta) \quad (9)$$

Considering the system complexity, instead of calculating the gradient of $J(\theta)$ directly, exploiting the Monte Carlo method [10], $\nabla_\theta J(\theta)$ can be approximated with a random gradient given an instance pair $s_t, a_t$. Then, the equation (9) can be updated to (10).

$$\theta_{new}\leftarrow\theta+\eta\cdot Q_\pi(s_t,a_t)\cdot\nabla_\theta\ln[\pi(a_t|s_t;\theta)] \quad (10)$$

As equation (11) reveals, the Monte Carlo method is also utilized for approximating the action-value function $Q_\pi(s_t,a_t)$ with a group of complete observations $s_t,\dots,s_M,a_t,\dots,a_M$ called training record. Therefore, the PPAP neural network learns to control the particle filter using the training records.

$$\theta_{new}\leftarrow\theta+\eta\cdot U_t(s_t,\dots,s_M,a_t,\dots,a_M)\cdot\nabla_\theta\ln[\pi(a_t|s_t;\theta)] \quad (11)$$

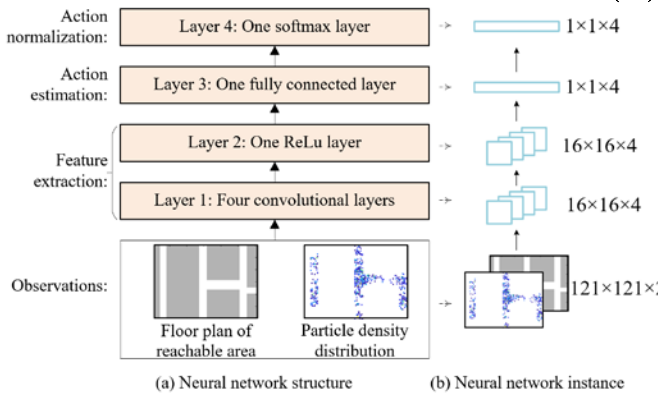

Action normalization:
Layer 4: One softmax layer
Action estimation:
Layer 3: One fully connected layer
Feature extraction:
Layer 2: One ReLu layer
Layer 1: Four convolutional layers
Observations:
Floor plan of reachable area
Particle density distribution
1×1×4
1×1×4
16×16×4
16×16×4
121×121×2
(a) Neural network structure
(b) Neural network instance

Fig. 2 Structure of the proposed neural network for the PPAP.

The structure of the proposed PPAP neural network is shown in Fig. 2 (a). As an example, the reachable area around the positioning result and view of the particle distribution has been extracted as the observation. Then, a convolution layer and a ReLu (rectified linear unit) layer extract features from the observation. Successively, a fully connected layer estimates the probabilities of each action with different divergence values. Finally, the softmax layer normalizes the probability of the different actions. Fig. 2 (b) reveals an instance of the proposed PPAP neural network. Our system randomly samples an action based on the action probabilities, then updates the particle filter.

## IV. Implementation and Results Evaluation

We have conducted our experiments at an open space office on the 7th floor of a research building. The environment contains several iron file cabinets and pillars, thus orientation based on the geomagnetic field is differently distorted at different places. Seven volunteers—2 females and 5 males, ages from 22 to 35—were invited to hold a smartphone and naturally walk within the testbed at normal speed. The smartphone pointing direction was in accordance with the user moving direction. During the sampling, the ground truth was inputted by the volunteers through a smartphone application. More than 1000 ground truth points have been inputted by the volunteers and about 300,000 have been elaborated.

The sampled trajectories have been equally divided into two parts based on their position in the testbed: west area and east area. In the west area, about 120,000 samples are used for training, and 30,000 samples are used for trained area testing. The east area is used for testing into an unknown area and contains about 150,000 samples.

### A. *Neural network training*

In this section, we detail an experiment for testing the training process of the neural network. As Fig. 3 (c) reveals, the reduced return of the system varies significantly at the beginning of the training epochs, then it converges with high return values (e.g., around 65 epochs), and finally it diverges again after 80 epochs. During the training phase, in order to prevent that the agent sticks into local optima, the system leverages entropy loss weight to penalize actions with high certainty. Basically, the action taken in the training phase is added with randomness to explore more possible action combinations. Therefore, the return at beginning epochs can only occasionally reach a high return value. Around 65 epochs, the policy neural network reaches the optimal setting. Then, the system return is stable to a high value. Therefore, we export the neural work at this epoch as the policy model. It is worth noting that if the training continues, the return may drop again because the exploration mechanism leads the neural network to leave the optimal state.

The experiment also examines the influence of different particle numbers in the training of the neural network. As Fig. 3 (a) and (b) reveals, we observe the reduced return of particle filters with 300 and 50 particles for the first 100 epochs. Although the running time for every epoch drops, the system becomes hard to converge because the particle filter requires enough particles to represent the probability distribution of the user position. In the office scenario, used as a testing scenario, 1000 particles are enough to depict the position probability.

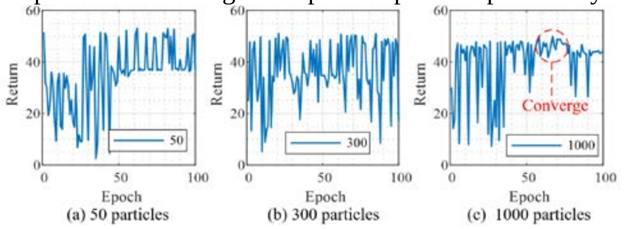
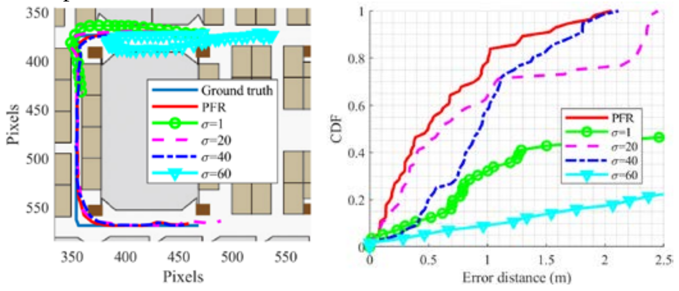


Fig. 3 The return variation of different particle counts with the increasing of training epochs.

## B. *Adaptive action analysis*

As shown in Fig. 2 (b), our system leverages an observation window of $121 \times 121$ pixels. The scale of our system is 11.94 pixels per meter, thus the observation window has a length of 10.13m. The action space contains four standard deviations of moving direction, including 1, 20, 40, 60. The PFR utilizes 1000 particles.



(a) Positioning traces (b) Positioning accuracy

Fig. 4 Positioning trace and accuracy of a single road. The ticks of the X and Y axis in figure (a) are pixels.

It is worth noting the importance of keeping a proper particle divergence. On the one hand, too low divergences lead the particles to lose track, especially when the observations contain several errors. On the other hand, too high divergences decrease the positioning accuracy. Particle divergence is an effective method to control the particle distribution and we propose the PPAP neural network to automatically estimate the moving direction divergence based on the state of the particle distribution.

The experiment compares the algorithm with the traditional choice to consider a constant direction of deviation. Considering that our system adopts four standard deviation values, the experiment tests the positioning performance varying the action value to these four constant values. Fig. 4(a) reveals the evaluated positioning traces. When the action is equal to 1 or 60, the particles lost the target after few steps. In the $\sigma = 1$ case, all the particles hit walls and die due to the particle divergence set. In the $\sigma = 60$ case, particles scatter everywhere and lost the target due to the limited particles available. As shown in Fig. 4 (b), although the $\sigma = 20$ and $\sigma = 40$ cases allowed to follow the target along the whole path, the proposed PFR method outperforms the two cases because of the dynamic selection of optimal action values.

In order to analyze the efficiency of the PFR algorithm, we detail the actions at every step, as shown in Fig. 5 (b). Two action values are taken in the trace, 20 and 60. It can be found that most of the actions along the trace are 20, indicating a low orientation distortion. Particle distribution in these areas is tight as shown in Fig. 5 (a), thus improving the positioning accuracy. On the other hand, the PFR algorithm executes action equal to 60 when users approach corners. Due to the presence of pillars, which contain iron, we observe a severe distortion of geomagnetic orientations. Consequently, the PFR algorithm increases the orientation deviation to improve system robustness. As obs4, obs5, and obs6 in Fig. 5 (c) reveal, particles become sparse to cover more area. Although these actions decrease the accuracy, they improve the possibility of covering the ground truth position, thus preventing the end of tracking and increasing future accuracy.

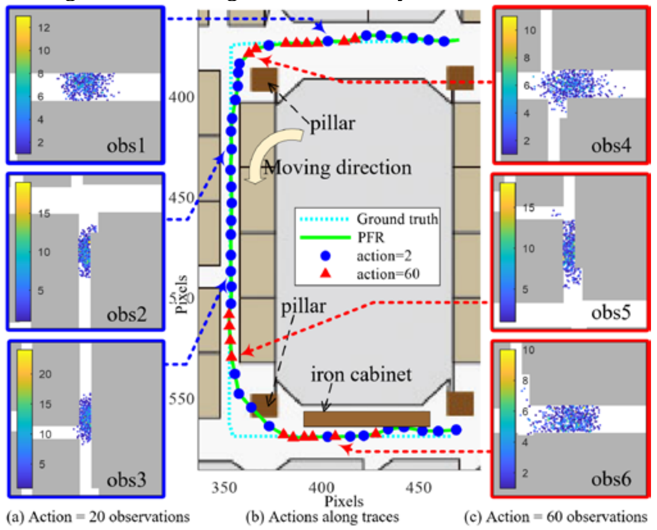


Fig. 5 Actions and observations along a positioning trace (Color bar indicates the number of particles at the point).

## C. *Direction bias vs. direction variance estimation*

This experiment compares performances of estimating the direction distribution variance with the performances obtained by estimating the adjustment to the direction. We replace the layer 3 in Fig. 2 with a fully connected network with only one element as output. In order to prevent the direction cyclicity problem, a tanh layer is appended to confine the network output in the scope of (-1, 1). Finally, a linear scaling layer transforms the direction compensation scope to (-90, 90) degrees. We set the direction variance to constant values of $\sigma = 0, 5, 10, 20$ degrees respectively.

Fig. 6 reveals the smoothed return value of the first 220 training epochs. All the cases tested have reached their maximum values. The pure direction drift estimation ($\sigma = 0$) is the worst. Adding a direction variance helps in increasing positioning accuracy, but the performance is lower than the proposed variance estimation method. In fact, the direction biases of the particles vary, especially in turning areas or approximating ferromagnetic furniture. The proposed method estimates the direction variance of the particles, therefore outperforms the direction drift estimation method.

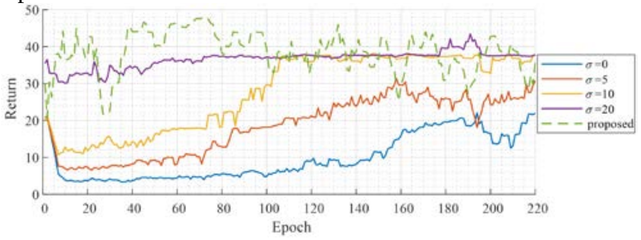


Fig. 6 Training performance comparison of continuous and discrete policies.

### *D. Test in known and unknown areas*

In order to test the model adaptability, we have conducted experiments in known and unknown scenarios. The known scenario is an area where a sampler has collected the walking traces and the collected data are used to train the neural network. Finally, new data are collected by a user to test the maximum performance of the system. The unknown scenario is an area that the model has not seen before and the data collected are used to test the model adaptability. The test compares the performance of the proposed adaptive algorithm and two particle-filters with constant standard deviations of the direction setting to $\sigma = 20$ and $\sigma = 40$ respectively. These two values were the best values obtained in the adaptive action analysis test.

Fig. 7 (a) shows the performance of the known area. Our proposal outperforms the traditional constant variance algorithm improving the overall positioning accuracy. It is worth noting that although the $\sigma = 20$ case has more results of errors less than one meter, the risk of losing the target increases due to the errors. The phenomenon reveals the importance of balancing accuracy and robustness. Into the unknown area, we test the trained model on new paths that the model has not seen before. Performances, shown in Fig. 7 (b), drop due to the unseen and not previously learned path. However, the system performance is still promising because the system is also able to estimate deviations of the orientation through the particle distribution. Furthermore, the PFR method still outperforms the traditional particle filters based on constant values. Although the $\sigma = 20$ and $\sigma = 40$ parameters are optimal in the trained scenario, they might be not suitable in new environments.

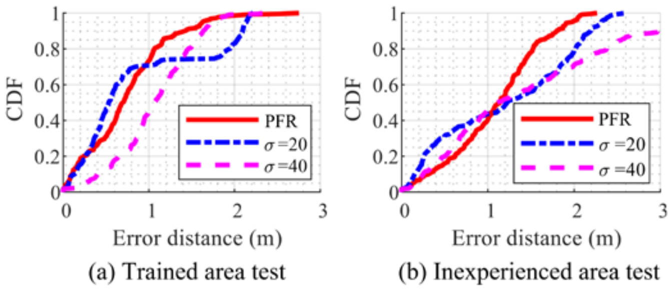


Fig. 7 Performance comparison of known and unknown paths.

### *E. Comparison with traditional indoor PDR algorithms*

The proposed algorithm is applicable in most of the particle-filter-based PDR methods. Therefore, we compare the performances of several state-of-the-art PDR and the proposed PFR. Direction filter [4] leverages the direction bias calculated in the training phase to compensate for real-time measurements, but it is limited by the sampling density. MaLoc [11] utilizes first-time compass direction and gyroscope increment to reduce the influence of the distorted geomagnetic field, but it suffers from the gyroscope drift problem. Wi-Fi based methods [12, 13] periodically add particles near ground-truth positions to keep particles in tracking. Fig. 8 reveals the comparison results showing that traditional direction compensation methods are helpful in improving PDR accuracy. The proposed PFR algorithm further enables traditional particle filters by adaptively adjusting real-time direction variance according to particle distributions and nearby floor plans, therefore it achieves higher overall accuracy.

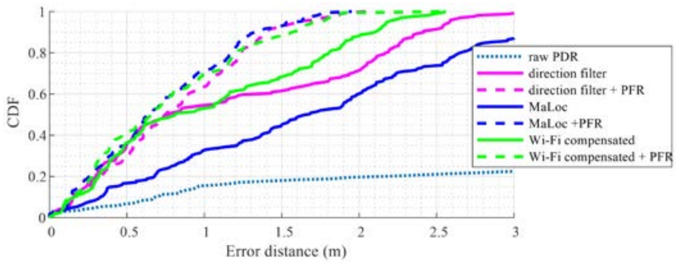


Fig. 8 Performance comparison with traditional PDR algorithms.

## V. Conclusion

The paper presents the PFR, a novel particle filter based on reinforcement learning techniques. Compare to conventional particle filters, experiments performed in real-world scenarios reveal the system adaptability in estimating a proper direction variance when user walk and performances improve more than 20% at the 80% probability compared with state-of-the-art methods. As future works, we should study how to leverage correlations across the entire maps, investigating potential optimal ways to represent particles and researching adaptability in complex buildings with multiple sources of information.

## References

[1] L. Zhang, M. Cheng, Z. Xiao, L. Zhou, and J. Zhou, "Adaptable map matching using PF-net for pedestrian indoor localization," *Ieee Commun Lett,* vol. 24, no. 7, pp. 1437-1440, 2020.

[2] W. Shao, H. Luo, F. Zhao, C. Wang, A. Crivello, and M. Z. Tunio, "DePedo: Anti Periodic Negative-Step Movement Pedometer with Deep Convolutional Neural Networks," in *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, 20-24 May 2018 2018, pp. 1-6, doi: 10.1109/ICC.2018.8422308.

[3] Q. Wang, L. Ye, H. Luo, A. Men, F. Zhao, and C. Ou, "Pedestrian walking distance estimation based on smartphone mode recognition," *Remote Sensing,* vol. 11, no. 9, p. 1140, 2019.

[4] W. Shao, H. Luo, F. Zhao, C. Wang, A. Crivello, and M. Z. Tunio, "Mass-centered weight update scheme for particle filter based indoor pedestrian positioning," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Spain, 15-18 April 2018 2018, pp. 1-6, doi: 10.1109/WCNC.2018.8377274.

[5] J. L. Carrera Villacres, Z. Zhao, T. Braun, and Z. Li, "A Particle Filter-Based Reinforcement Learning Approach for Reliable Wireless Indoor Positioning," *IEEE Journal on Selected Areas in Communications,* vol. 37, no. 11, pp. 2457-2473, 2019, doi: 10.1109/jsac.2019.2933886.

[6] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature,* vol. 550, no. 7676, pp. 354-359, 2017, doi: 10.1038/nature24270.

[7] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *Ieee T Signal Proces,* vol. 50, no. 2, pp. 174-188, 2002.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[9] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *NIPs*, 1999, vol. 99: Citeseer, pp. 1057-1063.

[10] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American statistical association,* vol. 44, no. 247, pp. 335-341, 1949.

[11] H. Xie, T. Gu, X. Tao, H. Ye, and J. Lv, "MaLoc: A Practical Magnetic Fingerprinting Approach to Indoor Localization using Smartphones," presented at the UBICOMP, SEATTLE, WA, USA, SEPTEMBER 13 - 17, 2014, 2014.

[12] W. Shao, H. Luo, F. Zhao, and A. Crivello, "Toward improving indoor magnetic field–based positioning system using pedestrian motion models," *Int J Distrib Sens N,* vol. 14, no. 9, p. 1550147718803072, 2018, doi: 10.1177/1550147718803072.

[13] W. Shao, H. Luo, F. Zhao, H. Tian, J. Huang, and A. Crivello, "Floor Identification in Large-Scale Environments with Wi-Fi Autonomous Block Models," *IEEE Transactions on Industrial Informatics,* pp. 1-1, 2021, doi: 10.1109/TII.2021.3074153.