Clearing a crowd: Context-supported Neighbor Positioning for People-centric Navigation

Takamasa Higuchi[†], Hirozumi Yamaguchi[†,‡], and Teruo Higashino[†,‡]

[†] Graduate School of Information Science and Technology, Osaka University, Japan
1-5 Yamadaoka, Suita, Osaka 565-0871 Japan
[‡] Japan Science and Technology Agency, CREST
{t-higuti,h-yamagu,higashino}@ist.osaka-u.ac.jp

Abstract. This paper presents a positioning system for "people-centric" navigation, which estimates relative positions of surrounding people to help users to find a target person in a crowd of neighbors. Our system, called PCN, employs pedestrian dead reckoning (PDR) and proximity sensing with Bluetooth only using off-the-shelf mobile phones. Utilizing the feature of "group activity" where people naturally form groups moving similarly and together in exhibitions, parties and so on, PCN corrects deviation of distance and direction in PDR. The group information is also helpful to identify the surrounding people in the navigation. A field experiment in a real exhibition with 20 examinees carrying Google Android phones was conducted to show its effectiveness.

1 Introduction

Let us think of a party place as shown in Fig. 1. Since the place is highly crowded with those present, our view is often obstructed by the surrounding people as Fig. 2. Thus we can hardly find a particular person in such a crowd even if we know that he/she is nearby. Unfortunately, there is few technology that can help us in such situations. Infrastructure-based location systems using ultrasound, infrared or RF usually need a lot of embedded sensors on the walls and dedicated receivers at clients. WiFi-based systems [3, 16] are most popular, but accuracy is greatly affected by noise from the other APs. The related indoor positioning methods are surveyed in Section 2. We strongly believe that such navigation should be achieved instantly and easily without relying on special equipment. Use of off-the-shelf devices such as smartphones is a reasonable option, but to mitigate errors due to sensor noise and other environmental factors is a challenge.

This paper presents a positioning system called *PCN* (the acronym of people-centric navigation) that estimates the relative positions of surrounding people to help users to find a specific person in the crowd of neighbors. PCN employs pedestrian dead reckoning (PDR) and proximity sensing with Bluetooth only using off-the-shelf mobile phones. As demonstrated in our preliminary experiment being presented in Section 3, position errors in PDR may grow up to tens of meters, which may seriously degrade the relative position accuracy. To cope with the problem, we focus on *group activity*, a specific feature in crowded situations

Fig. 1. A scene of a party

Fig. 2. Finding a person in a crowd

like exhibitions and parties. In such situations, people often move together with some others (*i.e.*, "groups"). These groups may be formed by friends, families and colleagues, or even strangers who are just moving toward the same direction. PCN dynamically detects similarity of their activities by gathering mobile phones' acceleration, direction and received signal strength (RSS) of Bluetooth radios, and then corrects deviation of PDR trace by harmonizing with the traces of other group members. The estimated group information is also helpful for enhancing position awareness of the users, which is usually based on recognition of groups like "three people walking ahead" and "five people standing behind".

To quantitatively observe the performance in terms of group recognition precision and relative position accuracy, we have implemented PCN on Google Android phones and conducted a field experiment with 20 examinees in a real exhibition. The result of the experiment has shown that PCN could achieve group estimation accuracy of 94.8% as well as relative position accuracy of 3.51m. Without depending on any additional infrastructure or dedicated devices, PCN has reduced the position error by 31% compared to a conventional approach.

2 Related Work and Contribution

A number of methods have been investigated to estimate location of mobile nodes. Active Bat [8] is an well-known method that employs TDoA (time difference of arrival) of ultrasound and radio signals to measure the distance between clients and multiple anchors. Taking the advantage of accurate range measurement by TDoA, positioning errors are usually within a few centimeters if the anchors are deployed with a few meters spacing. However, the cost for installation and maintenance of such anchors is not negligible. We have proposed a cooperative localization method called "stop-and-go localization" [9] for mobile nodes with TDoA ranging devices, which achieves accurate positioning with much less dependence on infrastructure. However, accurate distance measurement between commercial mobile devices is still a challenge. Although there are some techniques like BeepBeep [12] that achieves accurate ranging between mobile phones using propagation delay of sound signals, additional effort such as device-dependent tuning and background sound noise elimination is necessary and thus more instant and simpler positioning is preferable. For more cost-efficient positioning, Virtual Compass [1] employs wireless ad hoc communication via Wi-Fi and Bluetooth to estimate relative distance between mobile nodes based on RSS. However, RSS-based ranging often incurs large error due to multipath or other signal propagation effects.

Pedestrian dead reckoning [13, 15] that estimates the trace of pedestrians using accelerometers, digital compasses and gyro sensors has been investigated so far. While most previous PDR methods have assumed dedicated sensor devices attached to human bodies, some methods such as CompAcc [6] have utilized commercial mobile phones. However, due to noise from the sensors, PDR cannot stand alone as demonstrated in the next section.

Utilizing proximity information is a cost effective way to refine the accuracy of estimated traces obtained by PDR or some other ways. We have presented an encounter-based trace estimation method in [7]. NearMe [11] detects proximity of mobile phones by comparing the list of detected Wi-Fi APs and their RSS. Escort [5] combines PDR with proximity sensing via sound beaconing to estimate relative positions, assuming such services that guide users to their friends in unknown places. Our approach is similar with Escort in the sense that we also use PDR and proximity information. However, both of our goal and approach are totally different from Escort since we pursue the estimation accuracy of neighbors' relative positions to identify the friends nearby, while Escort aims at guiding users to friends not in the surrounding crowd but in some unknown or unfamiliar places. Escort relies on image recognition to finally find friends in a crowd, which may incur additional effort by users. Similarly, [10] enhances the quality of PDR by detecting proximity information based on Bluetooth RSS, but the accuracy would still not be enough for such mobile social navigation.

Based on the discussion above, our contribution is two-fold. Firstly, we present a novel positioning method to identify nearby friends in a crowd only using off-the-shelf mobile phones. To the best of our knowledge, this is the first approach that copes with this challenge. Considering the feature of people's behavior in exhibitions, parties and so on, we have come up with the idea of detecting "group activity" from sensor readings and Bluetooth of mobile phones and utilizing it to correct relative position errors. Secondly, we have implemented our system on Google Android phones, and have conducted experiments in a real exhibition. Through this experiment, we could show the effectiveness of our approach in the real world.

3 Preliminary Experiments and Basic Idea

The overview of PCN system is shown in Fig. 3. Clients of PCN (*i.e.*, mobile phones) continuously obtain accelerometer and digital compass readings to estimate step counts and direction. They also estimate a vector of each step called *step vector*, using the direction information and stride length. Since the stride length varies between individuals, it is approximated from the body height. The clients also record RSS from the neighboring clients, which is collected through device discovery process of Bluetooth. The step vectors and RSS are transferred to a centralized server called *PCN server* via 3G or Wi-Fi, and the collected RSS

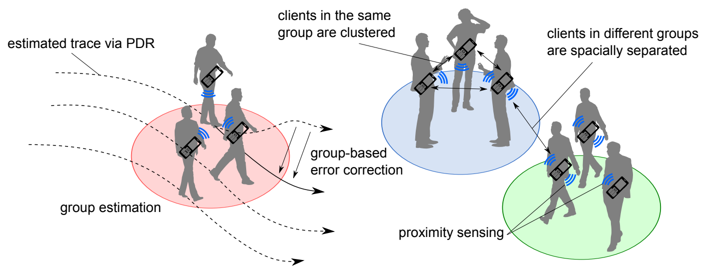
estimated trace via PDR
clients in the same group are clustered
clients in different groups are spacially separated
group-based error correction
group estimation
proximity sensing

Fig. 3. PCN system overview

is transformed into distance based on a predefined RSS-to-distance function at server side. Then the PCN server estimates relative positions among users and the results are sent back to the clients to tell the estimated situation.

3.1 Effect of Noise on Proximity and Trace Estimation

As we have stated in the previous sections, step vectors and RSS-based distance contain non-negligible errors. Fig. 4 shows Bluetooth RSS-distance mapping based on a real measurement using two Google Android phones (Samsung Nexus S) in our department building receiving a lot of interference from Wi-Fi. We have plotted the measured RSS at each distance (outliers have been eliminated), and indicated their maximum and average values, where error bars show the standard deviations from the average. As shown in previous literature such as [4], we can see that different RSS values were observed at the same distance due to multipath effect, interference and so on. However, we focus on a criterion to characterize this relation based on the maximum RSS values; at 7m or longer distances they never reach -70dBm, while they exceed it at 6m or shorter distances. We utilize this characteristic to detect "proximity" explained later, allowing some inaccuracy around the boundary of two categories.

We have also implemented a simple PDR application on the Android phones to examine the accuracy of step vectors. This application continuously monitors acceleration in the vertical direction to detect steps and the compass readings to estimate the orientation of mobile phones. As shown in Fig. 5, the acceleration in the vertical direction changes synchronously with the user's steps. Therefore we simply count up the number of steps when the acceleration exceeds a threshold where the counting interval is set to 300 milliseconds to prevent double counting. Using this application, we have analyzed the estimated trace of a user walking twice on the boundary of a 5m$\times$10m rectangle region (Fig. 6). On the estimated trace, the true positions of the three different points highlighted by dotted circles are actually the same, and thus we can observe that the position errors grew up to 10.32m after 60m walking. We note that this is the simplest threshold-based PDR where mobile phones are assumed to be held vertically at hands. There are of course more enhanced methods such as [13], and those methods can be used to improve PDR accuracy in PCN since it just uses trace estimation results from PDR. For simplicity of discussion, we assume this simple PDR hereafter.

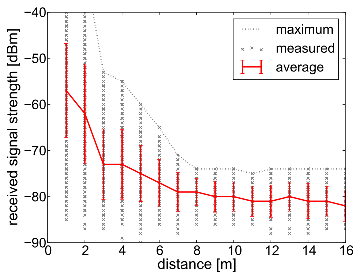

Fig. 4. Distance vs. Bluetooth RSS

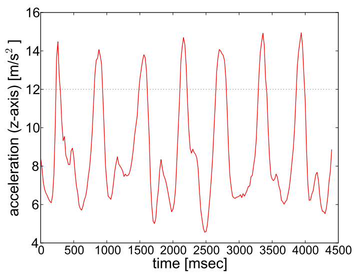

Fig. 5. Acceleration in vertical direction

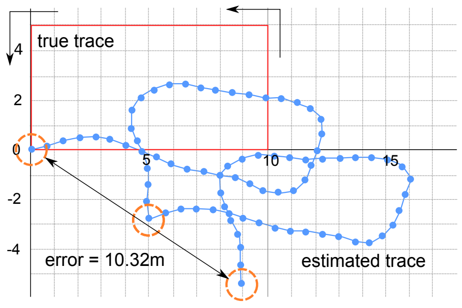

Fig. 6. Trace deviation in PDR

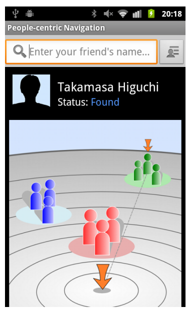

Fig. 7. Prototype of PCN client

3.2 Preliminary Experiment for Context-Supported Positioning

In order to calibrate PDR traces, we focus on general observation of human behavior; in crowded regions, the behavior can be categorized into several patterns. For example, in a party, most people stand and talk with each other. They often move around together to join the other groups or to find drinks and foods. To examine similarity of traces in a group, we have conducted the following experiment using the PDR application. We let 15 examinees with Nexus S phones freely form groups and let them walk for 30 minutes in a 10m$\times$10m field where markers were placed with one meter spacing. The examinees also took videos of markers to record true traces as shown in Fig. 8. The obtained traces were broken down into subtraces of 2 sec., and the average direction of each group was derived at each time. Then directions of the subtraces were compared to each group average to examine the deviation of orientation within and without the group. Fig. 9 shows the cumulative distribution of directional deviation from the group average. The deviation was 30 degrees or less for about 80% of subtraces in the same group, while it distributed uniformly for those in different groups.

Assuming that users in the same group have similarity of traces, PCN corrects the estimated traces that are deviated from the "group traces". Since the group information is estimated from the collected acceleration, direction and Bluetooth RSS, we do not need any additional information for the error correction. We define two criteria to recognize groups; nodes (mobile phones) currently in the same group (i) have been close to each other, and (ii) have moved simi-

Fig. 8. Preliminary experiment

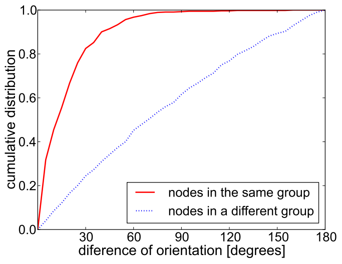

Fig. 9. Deviation of moving directions

larly for some duration. Properties (i) and (ii) are called **proximity** and **trace similarity**, respectively. Based on these properties, *group likelihood* is calculated and the most likely grouping is adopted.

The estimation process of PCN is as follows. When the current traces are reported from the clients to the PCN server, deviated directions and positions are calibrated based on the trace similarity and proximity properties, respectively. For this purpose, groups are estimated using the past acceleration, direction and Bluetooth RSS information. After that, the Bluetooth RSS values are utilized to reflect the relative distance among the nodes.

Fig. 7 shows a screenshot from our PCN client on the Android platform. If group estimation is correct, the user can identify the target person in the group of three people behind the group of three people in front of him/her. This context information helps to mitigate the bad effect of position errors in recognizing the target person.

4 System Design

Let S denote the set of PCN clients. PCN server estimates relative positions of these clients using Bluetooth RSS and user traces reported by each client $A_i \in S$. Estimated trace via PDR is obtained as a sequence of 2-dimensional step vectors $\boldsymbol{s}_k$, each of which represents the displacement by a step. For every τ sec., we define a *movement vector* of each A_i by the total displacement over the last τ sec. We denote the movement vector of A_i at time t by $\boldsymbol{u}_{i,t}$ in the algorithm descriptions below. As well, we let $r_{ij,t}$ denote measured RSS between A_i and $A_j \in S$ at time t.

4.1 Proximity Sensing via Bluetooth Scans

First, we describe the design of proximity sensing. As mentioned above, PCN clients collect Bluetooth RSS from nearby clients via device discovery process, or Bluetooth scan. Since Bluetooth does not have explicit broadcast mechanism, it is the only way to instantly collect peer-to-peer RSS without link management.

Ordinary Android phones basically take more than 10 sec. for each attempt of Bluetooth scans, which imposes a severe limitation on the frequency of the scan attempts. To make matters worse, while the process of a scan, nodes hardly respond to the inquiries from other neighbors because they quickly change their radio frequency meanwhile. Consequently, nodes often miss nearby nodes if they just repeat Bluetooth scans.

Although the scan mechanism of Bluetooth is designed to robustly detect all the devices in the radio range, nearby nodes are relatively free from influence of unexpected signal attenuation, and usually respond more quickly to the inquiries. In proximity sensing, quickly detecting the neighbors in close proximity is more important than ensuring exhaustive detection. For that reason, we decided to interrupt each Bluetooth scan in 5 sec. from the beginning. As a result of preliminary experiment, we confirmed that the success rate of scans between nearby nodes within 5m is degraded by only about 10% by such interruption. In addition, we randomly determine the timing of scans to avoid misdetection caused by simultaneous scan attempts. At each time, nodes start scanning with probability of p_{scan}, or otherwise they sleep for a designated backoff time. In our implementation, we set p_{scan} to 0.5 and randomly pick out backoff time from 2.5 sec. to 7.5 sec. Thus, we achieve reasonable performance in proximity sensing only using Bluetooth.

4.2 Extracting Group Activities

PCN extracts group activities using recent user trails and measured RSS. These data are periodically reported by each client and maintained in the form of fixed length sequences. For each client A_i and time t, let $R_{ij}(t) = \{r_{ij,t}, r_{ij,t-\tau}, \ldots, r_{ij,t-(N-1)\tau}\}$ be the sequence consisting of the last N samplings of RSS from A_j, and $U_i(t) = \{\boldsymbol{u}_{i,t}, \boldsymbol{u}_{i,t-\tau}, \ldots, \boldsymbol{u}_{i,t-(N-1)\tau}\}$ be the sequence of its recent N movement vectors. In this paper, we set the unit time τ to 2.0 sec. and the window size N to 30, respectively.

We characterize similarity of activity (*i.e.*, group) by two measures; (i) **proximity** and (ii) **trace-similarity**. In proximity property, people in the same group are assumed to be continuously close to each other. For each pair $A_i, A_j \in S$ of clients, we quantify this feature by the number of recent Bluetooth scans with RSS that exceeds a threshold Θ_{prox}:

$$n_{ij}(t) = |\{r_{ij,t'} | r_{ij,t'} \in R_{ij}(t), r_{ij,t'} > \Theta_{prox}\}| \quad (1)$$

where Θ_{prox} is given as a system parameter. In trace-similarity property, the clients in the same group are assumed to have traces that are spatiotemporally similar to each other. To quantify this feature, we extend edit distance on real sequences (EDR)[2], a similarity measure for trajectories which was originally designed for similarity-based retrieval on a trajectory database. For each pair $A_i, A_j \in S$ of clients, we define the edit distance between sequences of their recent movement vectors. In general, edit distance between two sequences A and B is defined by the number of insert, delete, and replace operations that are needed to convert A into B. We regard corresponding movement vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ to be matched if they satisfy the both of following criteria:

$$|\ ||\boldsymbol{u}|| - ||\boldsymbol{v}||\ | < \Theta_l, \quad |\arg(\boldsymbol{u}) - \arg(\boldsymbol{v})| < \Theta_\theta \quad (2)$$

where Θ_l and Θ_θ are matching thresholds. Note that we skip the angular criteria if $\boldsymbol{u} = \boldsymbol{0}$ or $\boldsymbol{v} = \boldsymbol{0}$ since we cannot define orientation of the vector in such cases. Let U and V be the sequences of movement vectors, where the length of each sequence is n and m, respectively. We define the edit distance between U and V by the following recursive formula:

$$ED(U,V) = \begin{cases} n \cdot w & \text{if } m = 0 \\ m \cdot w & \text{if } n = 0 \\ \min\{ED(Rest(U), Rest(V)) + c, & \\ \qquad ED(Rest(U), V) + w, & \\ \qquad ED(U, Rest(V)) + w\} & \textit{otherwise} \end{cases} \tag{3}$$

where $Rest(U)$ represents a sub-sequence of U without the first element. c is a cost function of a replace operation where $c = 0$ if the first element of U and that of V satisfy the matching criteria, and $c = 1$ otherwise. Insert and delete operations correspond to temporal shifting on user traces. We let w denote the cost of such a shifting operation, and set it to 0.5. There could be a small difference in timing of movements even if A_i and A_j belong to the same group. We intend to mitigate the impact of such temporal variations by imposing a smaller cost value to insert and delete operations.

We define the edit distance $d_{ij}(t)$ between the sequences of recent movement vectors for each client pair A_i and A_j, say $U_i(t)$ and $U_j(t)$, as follows:

$$d_{ij}(t) = ED(U_i(t), U_j(t)) \tag{4}$$

For short, we denote $n_{ij}(t)$ and $d_{ij}(t)$ by just n_{ij} and d_{ij} in the following descriptions. In PCN, clients that satisfy both proximity and trace-similarity each other are regarded as a group. We represent the group relationship between clients A_i and A_j by binary random variable G_{ij}, where $G_{ij} = 1$ if they belong to the same group and $G_{ij} = 0$ otherwise.

Probability distribution of G_{ij} under given measurements of n_{ij} and d_{ij} can be derived by the well-known Bayes' rule:

$$P(G_{ij}|n_{ij}, d_{ij}) = \frac{P(n_{ij}, d_{ij}|G_{ij}) \cdot P(G_{ij})}{\sum_{G_{ij}=0}^{1} P(n_{ij}, d_{ij}|G_{ij}) \cdot P(G_{ij})} \tag{5}$$

For simplicity, we assume that n_{ij} and d_{ij} are independent under given G_{ij}:

$$P(n_{ij}, d_{ij}|G_{ij}) = P(n_{ij}|G_{ij}) \cdot P(d_{ij}|G_{ij}) \tag{6}$$

Here we also assume that there is no prior information about group relationship between A_i and A_j, that is, $P(G_{ij} = 0) = P(G_{ij} = 1) = 0.5$. Under these assumptions, Eq.(5) can be simplified as:

$$P(G_{ij}|n_{ij}, d_{ij}) = \frac{P(n_{ij}|G_{ij}) \cdot P(d_{ij}|G_{ij})}{\sum_{G_{ij}=0}^{1} P(n_{ij}|G_{ij}) \cdot P(d_{ij}|G_{ij})}. \tag{7}$$

Distributions $P(n_{ij}|G_{ij})$ and $P(d_{ij}|G_{ij})$ in Eq. (7) can be obtained by a prior learning, which will be discussed in Section 5.1. To classify clients $A_i \in S$ ($i =$

$1, 2, \dots, n$) into activity groups, for each pair of $A_i, A_j \in S$, we first calculate the probability that they belong to the same group based on n_{ij} and d_{ij}. Hereafter let us call the probability $P(G_{ij} = 1|n_{ij}, d_{ij})$ *group likelihood*.

Next, we construct a "grouping graph" as follows. Each client corresponds to a vertex of the graph, and we add an edge between A_i and $A_j \in S$ if and only if group likelihood $P(G_{ij} = 1|n_{ij}, d_{ij}) > \Theta_{group}$, where Θ_{group} is a threshold. In this paper, we set Θ_{group} to 0.5. Finally we regard each connected component on the grouping graph as an activity group.

4.3 Context-supported Relative Positioning

In this section, we describe the detailed design of our context-supported relative positioning. The estimation is completed through two phases; In the former phase, we correct user traces based on the trace-similarity of activity groups. Bending the trace within the limits of expected PDR error, we alleviate the impact of sensor noise. In the latter phase, we determine the positional relationship of user traces based on RSS with the help of proximity of activity groups. It is also effective to reduce position error, as well as to enhance the "perceptibility" of the resulting estimated positions.

Modeling PDR Error To correct user trace, PCN first estimates the expected error in the movement vectors. Assuming that PDR on client A_i has detected m steps during the last τ sec., the movement vector $\boldsymbol{u}_{i,t}$ can be denoted by $\boldsymbol{u}_{i,t} = \sum_{k=1}^{m} \boldsymbol{s}_k$, where $\boldsymbol{s}_k$ is the step vector by the k-th step. PDR error is mainly caused by i) error of step length estimation, ii) error of direction estimation, and iii) misdetection of user steps. Errors caused by i) and ii) occur in every step $\boldsymbol{s}_k$, and accumulated in the movement vector. Expected error of step $\boldsymbol{s}_k$ can be denoted as $\boldsymbol{e}_k = \boldsymbol{e}_{l_k} + \boldsymbol{e}_{\theta_k}$, where $\boldsymbol{e}_{l_k}$ is the error of estimated step length, say l, and $\boldsymbol{e}_{\theta_k}$ is the error caused by directional distortion (Fig. 10(a)). We assume that $\boldsymbol{e}_{l_k}$ is in the same direction as $\boldsymbol{s}_k$ and its length follows Gaussian distribution $\mathcal{N}(0, \sigma_l^2)$. For $\boldsymbol{e}_{\theta_k}$, we assume that direction estimation error $\Delta\theta$ follows Gaussian distribution $\mathcal{N}(0, \sigma_\theta^2)$, and approximate $\boldsymbol{e}_{\theta_k}$ by a vector which is orthogonal to $\boldsymbol{s}_k$ with a length of $l\Delta\theta$. Under this modeling, we can uniquely determine the error distribution of each step $\boldsymbol{s}_k$. We empirically set the model parameter σ_l and σ_θ to 0.5m and 30.0 degrees, respectively.

The error caused by iii) occurs regardless of the number of detected steps in the last τ sec. We denote this basic error by $\boldsymbol{e}_0$ and model it as follows:

$$P(\boldsymbol{e}_0) = \mathcal{N}(\boldsymbol{0}, \sigma_0^2 \boldsymbol{I}) \tag{8}$$

where $\boldsymbol{I}$ is a 2-dimensional unit matrix. We empirically set σ_0 to 1.0m.

Correcting User Traces Using the constructed step-level error prediction model, we estimate the probability distribution of a movement vector as follows. As mentioned above, PDR error is accumulated in the movement vector step-by-step as shown in Fig. 10 (b). Let $\boldsymbol{u}'_k$ be a partial movement vector composed

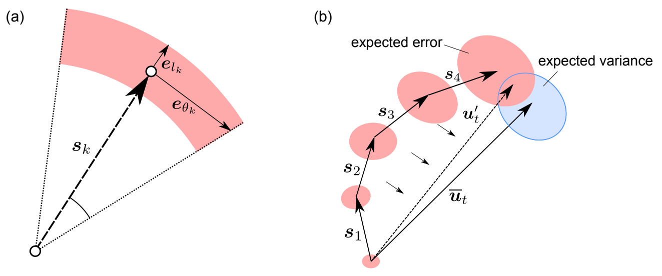

Fig. 10. Correcting user traces: (a) step-level error prediction model of PDR, (b) correcting a movement vector $\boldsymbol{u}_t$ according to the average vector $\overline{\boldsymbol{u}}_t$ of the group.

of $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_k$. We recursively predict the distribution of $\boldsymbol{u}_{i,t}$ by sequentially deriving the distributions of $\boldsymbol{u}'_k$ $(k = 0, 1, 2, \ldots, m)$, namely $P(\boldsymbol{u'}_k|\boldsymbol{s}_{1:k})$. Since the error caused by misdetection of user steps could occur regardless of the users' movement, we apply the distribution in Eq. (8) as an initial distribution:

$$P(\boldsymbol{u}'_0) = P(\boldsymbol{e}_0). \tag{9}$$

Given the distribution at the $(k-1)$-th step (*i.e.*, $P(\boldsymbol{u}'_{k-1}|\boldsymbol{s}_{1:k-1})$), we can derive the distribution at the next step using the step-level error prediction model. Relationship between $\boldsymbol{u}'_{k-1}$ and $\boldsymbol{u}'_k$ can be represented as follows:

$$\boldsymbol{u}'_k = \boldsymbol{u}'_{k-1} + \boldsymbol{s}_k + \boldsymbol{e}_k \tag{10}$$

where $\boldsymbol{e}_k$ is the error vector included in $\boldsymbol{s}_k$. According to the recursion formula, the updated distribution $P(\boldsymbol{u}'_k|\boldsymbol{s}_{1:k})$ can be derived from the previous distribution $P(\boldsymbol{u}'_{k-1}|\boldsymbol{s}_{1:k-1})$ and step-level error distribution $P(\boldsymbol{e}_k|\boldsymbol{s}_k)$:

$$P(\boldsymbol{u}'_k|\boldsymbol{s}_{1:k}) = \int \left\{ \int P(\boldsymbol{u}'_k|\boldsymbol{s}_k, \boldsymbol{e}_k, \boldsymbol{u}'_{k-1}) \cdot P(\boldsymbol{u}'_{k-1}|\boldsymbol{s}_{1:k-1}) d\boldsymbol{u}'_{k-1} \right\} \cdot P(\boldsymbol{e}_k|\boldsymbol{s}_k) d\boldsymbol{e}_k \tag{11}$$

Here, we sample N_p particles $\boldsymbol{u'}^{(j)}_{k-1}$ $(j = 1, 2, \ldots, N_p)$ from the previous distribution $P(\boldsymbol{u}'_{k-1}|\boldsymbol{s}_{1:k-1})$ to represent it by Monte Carlo Approximation:

$$P(\boldsymbol{u}'_{k-1}|\boldsymbol{s}_{1:k-1}) \simeq \frac{1}{N_p} \sum_{j=1}^{N_p} \delta(\boldsymbol{u}'_{k-1} - \boldsymbol{u'}^{(j)}_{k-1}) \tag{12}$$

By this particle-based representation, the updated distribution in Eq. (11) can be transformed into:

$$P(\boldsymbol{u}'_k|\boldsymbol{s}_{1:k}) = \frac{1}{N_p} \sum_{j=1}^{N_p} \left[\int P(\boldsymbol{u}'_k|\boldsymbol{s}_k, \boldsymbol{e}_k, \boldsymbol{u'}^{(j)}_{k-1}) \cdot P(\boldsymbol{e}_k|\boldsymbol{s}_k) d\boldsymbol{e}_k \right]. \tag{13}$$

For each particle $\boldsymbol{u'}^{(j)}_{k-1}$, we sample a single particle $\boldsymbol{e}^{(j)}_k$ from the step-level error distribution $P(\boldsymbol{e}_k|\boldsymbol{s}_k)$ to approximate it as follows:

$$P(\boldsymbol{e}_k|\boldsymbol{s}_k) \simeq \delta(\boldsymbol{e}_k - \boldsymbol{e}^{(j)}_k) \tag{14}$$

Note that we sample an error value *for each of N_p particles* to reasonably approximate the step-level error distribution $P(\boldsymbol{e}_k|\boldsymbol{s}_k)$ overall. By this approximation, Eq. (13) can be transformed into:

$$P(\boldsymbol{u}'_k|\boldsymbol{s}_{1:k}) \simeq \frac{1}{N_p}\sum_{j=1}^{N_p}\left[\int P(\boldsymbol{u}'_k|\boldsymbol{s}_k, \boldsymbol{e}_k, \boldsymbol{u}'^{(j)}_{k-1})\cdot\delta(\boldsymbol{e}_k - \boldsymbol{e}^{(j)}_k)d\boldsymbol{e}_k\right]$$
$$= \frac{1}{N_p}\sum_{j=1}^{N_p}\left[\delta\left(\boldsymbol{u}_k - (\boldsymbol{u}'^{(j)}_{k-1} + \boldsymbol{s}_k + \boldsymbol{e}^{(j)}_k)\right)\right] \tag{15}$$

Based on the discussion above, we design the error prediction algorithm as follows. First, we sample N_p particles $\boldsymbol{u}'^{(j)}_0$ $(j = 1, 2, \ldots, N_p)$ from $P(\boldsymbol{u}'_0)$ in Eq. (9). Then, for each step $\boldsymbol{s}_k$, we add $\boldsymbol{s}_k$ and $\boldsymbol{e}^{(j)}_k$, which is an error value sampled from $P(\boldsymbol{e}_k|\boldsymbol{s}_k)$, to each particle $\boldsymbol{u}'^{(j)}_k$. After repeating this operation for all the steps $\boldsymbol{s}_k$ in the period of recent τ sec., we can obtain the probability distribution $P(\boldsymbol{u}_{i,t})$ of the movement vector in the form of Monte Carlo representation as $P(\boldsymbol{u}_{i,t}) = P(\boldsymbol{u}'_m|\boldsymbol{s}_{1:m})$. We set N_p to 500 in our experiment in Section 5.

Assuming the trace-similarity in an activity group, we correct the current movement vector to make it approach the average movement vector of the group it belongs to. We achieve reasonable correction by harmonizing the predicted error distribution of PDR and expected distribution given by the assumption of trace-similarity among people in the same activity group. Let $\overline{\boldsymbol{u}}_t$ be the average movement vector of group G, which A_i of interest belongs to; $\overline{\boldsymbol{u}}_t = \frac{1}{|G|}\sum_{A_j \in G}\boldsymbol{u}_{j,t}$. The distribution of expected movement vector is modeled based on the preliminary experiment described in Section 3. For each movement vector $\boldsymbol{u}_{i,t}$ and its corresponding group average $\overline{\boldsymbol{u}}_t$, we compare the length and direction of $\boldsymbol{u}_{i,t}$ and $\overline{\boldsymbol{u}}_t$. To model the distribution, we assume that the ratio of $\max\left(\frac{||\boldsymbol{u}_{i,t}||}{||\overline{\boldsymbol{u}}_t||}, \frac{||\overline{\boldsymbol{u}}_t||}{||\boldsymbol{u}_{i,t}||}\right)$ follows a one-sided Gaussian distribution $\mathcal{N}(1.0, \sigma^2_{g_l})$ and $(\arg(\boldsymbol{u}_{i,t}) - \arg(\overline{\boldsymbol{u}}_t))$ follows $\mathcal{N}(0.0, \sigma^2_{g_\theta})$. Based on the result of our preliminary experiment, we set the parameters σ_{g_l} and σ_{g_θ} to be 0.50 and 30.0 degrees, respectively. Since we found that the distribution of $(||\boldsymbol{u}_{i,t}|| - ||\overline{\boldsymbol{u}}_t||)$ varies with the moving speed of users, here we employ the ratio of the norms, which is more robust against the users' movement, in modeling trace-similarity. By multiplying these two distributions, the distribution of expected movement vector under given group average $\overline{\boldsymbol{u}}_t$ can be derived. Weighting each particle $\boldsymbol{u}'^{(j)}_k$ with the prior distribution $P(\boldsymbol{u}_{i,t}|\overline{\boldsymbol{u}}_t)$, we calculate the expected value of $\boldsymbol{u}_{i,t}$, which is to be the corrected movement vector $\tilde{\boldsymbol{u}}_{i,t}$:

$$\tilde{\boldsymbol{u}}_{i,t} = \sum_{j=1}^{N_p}\boldsymbol{u}^{(j)}_{i,t}\cdot\frac{P(\boldsymbol{u}^{(j)}_{i,t}|\overline{\boldsymbol{u}}_t)}{\sum_{k=1}^{N_p}P(\boldsymbol{u}^{(k)}_{i,t}|\overline{\boldsymbol{u}}_t)}. \tag{16}$$

If either $||\boldsymbol{u}_{i,t}||$ or $||\overline{\boldsymbol{u}}_t||$ is zero, we replace it by a sufficiently small value ($<< 1$) to approximate the corresponding weight. What we essentially do in Eq. (16) is to multiply two different distributions of $\boldsymbol{u}_{i,t}$ together, normalize the product so that it becomes a valid PDF, and compute the mean of the new PDF to derive the corrected movement vector.

Determining Relative Positions PCN determines relative positions of the clients based on the corrected movement vectors and measured RSS. If some external positioning infrastructure such as Wi-Fi APs is available, we can start the estimation from those approximate user positions; if not, we initially place each client on a virtual coordinate system at random. Then, at every τ sec., we update and refine these positions using the corrected movement vectors and measured RSS. For each client A_i, we first independently update its position by adding the corrected movement vector $\tilde{\boldsymbol{u}}_{i,t}$ to its previous position $\boldsymbol{p}_{i,t-1}$. We use the resulting position $\boldsymbol{p}_{i,t}^{(0)}$ as an initial estimation of A_i's position at time t, and adjust it based on peer-to-peer RSS by the following iterative algorithm.

At each iteration k, we put virtual attracting force $\boldsymbol{f}_{ij,t}^{(k)}$ between all the client pairs A_i, A_j which have observed RSS of more than Θ_{prox} during the last τ sec.:

$$\boldsymbol{f}_{ij,t}^{(k)} = \begin{cases} \kappa \cdot \max\left(0, ||\boldsymbol{p}_{j,t}^{(k-1)} - \boldsymbol{p}_{i,t}^{(k-1)}|| - d_p\right) \cdot \dfrac{\boldsymbol{p}_{j,t}^{(k-1)} - \boldsymbol{p}_{i,t}^{(k-1)}}{||\boldsymbol{p}_{j,t}^{(k-1)} - \boldsymbol{p}_{i,t}^{(k-1)}||} & \text{if } r_{ij,t} > \Theta_{prox} \\ \boldsymbol{0} & \text{otherwise} \end{cases} \tag{17}$$

where $\boldsymbol{p}_{i,t}^{(k-1)}$ and $\boldsymbol{p}_{j,t}^{(k-1)}$ are the estimated positions of A_i and A_j in the virtual coordinate system at the previous iteration $k-1$, and κ is an algorithm parameter that characterizes the strength of the force. d_p is the maximum range where RSS of more than Θ_{prox} can be observed ($d_p = 6.0$m in our preliminary experiment).

As discussed in Section 3, the large deviation of measured RSS imposes a limitation to proximity sensing; basically the only information we can obtain from the measured RSS is whether the distance between the nodes are within d_p. To pursue finer-grained positioning, we inspire some heuristics to the position estimation utilizing the assumption of proximity among members of an activity group. For client pairs in the same group, we replace d_p by d'_p ($d'_p < d_p$), where d'_p is also the possible range for the measured RSS. It leads estimated positions of the nodes in the same group to be closer, which is expected to enhance the positioning accuracy in most situations. We set d'_p to 3.0m in this paper.

In psychological science, it is said that human perceives objects in close proximity as a group[14]. Based on this observation, we slightly adjust the node positions to spatially separate each group. This could help users intuitively identify the estimated positions with actual surrounding people. We put this heuristics to the position estimation by adding weak force $\boldsymbol{f'}_{ij,t}^{(k)}$ between each client pairs:

$$\boldsymbol{f'}_{ij,t}^{(k)} = \begin{cases} \kappa' \cdot \max\left(0, ||\boldsymbol{p}_{j,t}^{(k-1)} - \boldsymbol{p}_{i,t}^{(k-1)}|| - d_{p'}\right) \cdot \dfrac{\boldsymbol{p}_{j,t}^{(k-1)} - \boldsymbol{p}_{i,t}^{(k-1)}}{||\boldsymbol{p}_{j,t}^{(k-1)} - \boldsymbol{p}_{i,t}^{(k-1)}||} & \text{if } G_{ij} = 1 \\ -\kappa' \cdot \max\left(0, d_{p'} - ||\boldsymbol{p}_{j,t}^{(k-1)} - \boldsymbol{p}_{i,t}^{(k-1)}||\right) \cdot \dfrac{\boldsymbol{p}_{j,t}^{(k-1)} - \boldsymbol{p}_{i,t}^{(k-1)}}{||\boldsymbol{p}_{j,t}^{(k-1)} - \boldsymbol{p}_{i,t}^{(k-1)}||} & \text{otherwise} \end{cases} \tag{18}$$

where κ' ($\kappa' << \kappa$) is a constant given as an algorithm parameter. Thus, we separate each group by adding attracting force between the pairs in the same group, whereas putting repulsive force to the pairs in different groups. Note that we choose κ' to be much smaller than κ to make the measurement-based attracting force $\boldsymbol{f'}^{(k)}_{ij,t}$ takes priority in determining the estimated positions, intending to maintain the consistency with RSS measurements.

Under the assumptions above, we update the "velocity" $\boldsymbol{v}_i^{(k)}$ of each node A_i according to the resultant force affecting to A_i:

$$\boldsymbol{v}_i^{(k)} = \alpha \left(\boldsymbol{v}_i^{(k-1)} + \sum_{A_j \in S \setminus \{A_i\}} \left(\boldsymbol{f}^{(k)}_{ij,t} + \boldsymbol{f'}^{(k)}_{ij,t} \right) \right) \tag{19}$$

where α is a dumping coefficient, which is given as an algorithm parameter. Note that $\boldsymbol{v}_i^{(0)} = (0, 0)$ for each client. Then we adjust the position of each client A_i depending on its *velocity*:

$$\boldsymbol{p}^{(k)}_{i,t} = \boldsymbol{p}^{(k-1)}_{i,t} + \boldsymbol{v}^{(k)}_i. \tag{20}$$

We repeat the operations above until the number of iteration k reaches the termination criterion k_{term} or total amount of *velocity* $|| \sum_{A_i \in S} \boldsymbol{v}_i^{(k)} ||$ converges to less than v_{term} to determine estimated positions $\boldsymbol{p}_{i,t}$ at time t.

5 Evaluation

To collect sensor data and RSS logs in real environment, we conducted a field experiment in a public trade fair. As a part of a technical event named Knowledge Capital Trial 2011 (`http://www.kmo-jp.com/en/`), the trade fair was held at a 27m×40m-sized hall as shown in Fig. 11. Totally 16 industrial companies and universities exhibited their state-of-the-art technology while thousands of visitors went around the booths.

We let 20 students hold Nexus S phones and asked to go around the event place with a group of four people. A sensing application equipped with PDR and proximity sensing was running on each phone to record users' traces and RSS logs. Each group entered the place at the entrance, and then looked around 6-12 booths for about 30 minutes. After that, they left there through the exit shown in Fig.11. Usually they stayed at each booth for 1-5 minutes on average. To collect ground truth data of user traces, we assigned an additional person to each group to plot their true positions on a field map with time stamps, as well as taking their photos at certain time intervals. After conducting such experiment three times, we collected real sensor data and RSS logs which are about 1,800 minutes long in total (90min. logs for 20 examinees). In the following evaluation, we used logs of 2 experiments as a learning dataset to construct group classifier, and the remaining one as a test dataset to examine the performance.

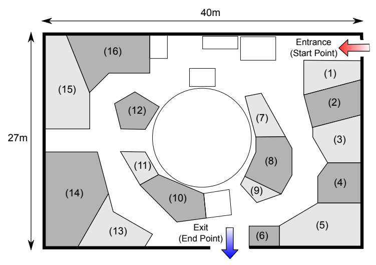

Fig. 11. Floor map

Fig. 12. Field experiment

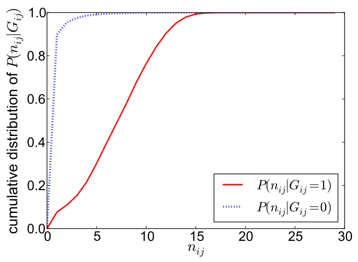

Fig. 13. Distribution of n_{ij}

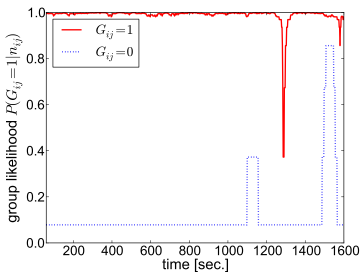

Fig. 14. Group likelihood (proximity)

5.1 Constructing Group Classifier

At first, we modeled proximity and trace-similarity of activity groups based on the learning dataset, and synthesized those models to construct a group classifier.

Proximity Model: Analyzing the learning dataset, we calculated the distribution of n_{ij}, which has been introduced as a proximity measure between two clients in Eq. (1). Fig. 13 shows the distribution in two different cases: one is for the pairs in the same group ($G_{ij} = 1$) and another is for ones in the different groups ($G_{ij} = 0$). As shown in the resulting distribution, $n_{ij} \leq 1$ in more than 90% cases for the pairs in different groups, while $n_{ij} \geq 2$ for as much as 95% for the same group cases. Thus, we can accurately distinguish whether or not a pair of clients belongs to the same group by observing the recent RSS measurements.

To examine the classification capability of the proximity feature, we tried constructing a group classifier only using the proximity model. We picked out RSS logs of three clients, say A_i, A_j, and $A_{j'}$ from the test dataset, where A_i and A_j belong to the same group while $A_{j'}$ is not. After applying the proximity-based group classifier to the pairs (A_i, A_j) and $(A_i, A_{j'})$, we obtained a series of group likelihood. The result is shown in Fig. 14. For the pair (A_i, A_j), the group likelihood is successfully around 1 throughout the experiment. As for $(A_i, A_{j'})$, the likelihood is less than 0.1 almost throughout the experiment. A problem is that the group likelihood of $(A_i, A_{j'})$ rises around $t = 1,100$ sec. and $t = 1,500$ sec. This is because these two groups were going around nearby booths at that time. The trace-similarity model, which will be discussed in the next section, helps to distinguish such nearby groups.

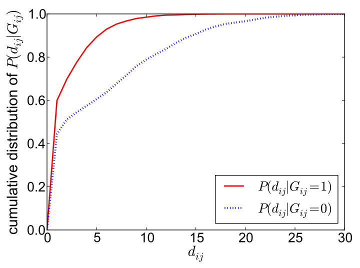

Fig. 15. Distribution of edit distance

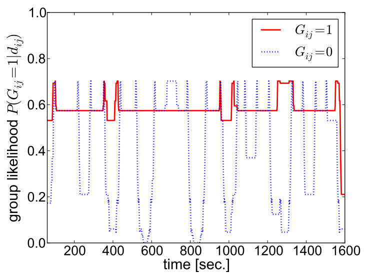

Fig. 16. Group likelihood (edit distance)

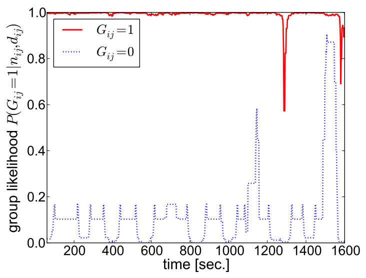

Fig. 17. Group likelihood (synthesized)

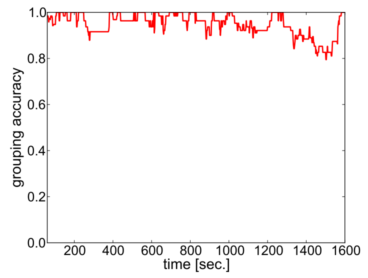

Fig. 18. Accuracy of group estimation

Trace-similarity Model: We calculated the distribution of d_{ij}, which has been defined as a trace-similarity measure in Eq. (4). The resulting distribution is shown in Fig. 15. Since we conducted this experiment in an exhibition, users spent much time staying at each booth. This leads the distribution of d_{ij} to be biased to zero. However, the probability that $d_{ij} \geq 5$ is around 40% for the clients in the same group while the corresponding probability for the different group cases is no more than 10%. This difference takes an important role in distinguishing nearby activity groups.

We also tried constructing a group classifier only using the trace-similarity model, and then applied the resulting classifier to the traces of pairs (A_i, A_j) and $(A_i, A_{j'})$, which correspond to the previous section. As a result, we obtained a series of group likelihood shown in Fig. 16. When the two clients are both staying at a booth, the group likelihood is relatively high regardless of whether they are in the same group or not. On the other hand, while the nodes travel between the booths, the likelihood in different group cases falls to around zero. The latter feature contributes to separation of nearby groups, as we mentioned above.

Group Classifier: Finally, we synthesize these two models using the Bayes' rule in Eq. (7) to construct a complete group classifier. Fig. 17 shows the group likelihood for pairs (A_i, A_j) and $(A_i, A_{j'})$ based on the resulting group classifier. Recalling the proximity-based group likelihood in Fig. 14, the likelihood of $(A_i, A_{j'})$ unsuccessfully rises when their groups are nearby. In Fig. 17, a failure around $t = 1,100$ sec. is mitigated since the large trace dissimilarity suppressed the group likelihood. Thus, the proximity and trace-similarity models complement each other to achieve better grouping accuracy. Note that if the nearby groups take similar behavior, these groups are regarded as the same group as $t = 1,500$ sec. in Fig. 17. Although this is a wrong estimation, we believe that it rarely affects the performance of PCN since users would also perceive such people as if they were in the same group.

5.2 Results

We evaluated the performance of PCN from three aspects, namely, grouping accuracy, relative positioning error, and perceptibility of the estimated positions.

Grouping Accuracy: Appropriate group classification is a key to enhance positioning performance with our context-supported correction mechanism. We applied the group classifier to all the sensor data and RSS logs in the test dataset to classify those 20 clients into activity groups. Then we evaluated the grouping accuracy by the accuracy rate of *pairwise membership test*: for each pair of nodes, we checked whether they are in the same group or not, and compared them with the actual grouping (5 groups of 4 people). As shown in the temporal change of resulting grouping accuracy in Fig. 18, we successfully achieved accuracy ratio of more than 90% over most pieces of time in the experiment, with average grouping accuracy of 94.8% (1.7% false positives and 3.5% false negatives). On the other hand, we can also see that the grouping accuracy temporarily drops around $t = 1,400$ sec. As discussed in Section 5.1, this is because several groups were staying at nearby booths, which leads the group classifier to misinterpret them as a single group. Although expanding the window size N to consider older histories of proximity and trace-similarity is a possible way to distinguish such groups, we should carefully select the parameter since larger N could make it hard to detect time variation of groups (*i.e.*, joining and leaving) immediately.

Relative Positioning Error: To evaluate the efficacy of our context-supported error correction mechanism, we evaluated relative positioning error to nearby nodes which are within 10m from each node A_i of interest. We represent the set of such nearby nodes at time t as $S_i(t) \subseteq S$ and define the average relative position error denoted by $err(t)$ as follows:

$$err(t) = \frac{1}{|S|} \sum_{A_i \in S} \left(\frac{1}{|S_i(t)|} \sum_{A_j \in S_i(t)} ||\boldsymbol{p}_{ji,t} - \tilde{\boldsymbol{p}}_{ji,t}|| \right) \qquad (21)$$

where $\boldsymbol{p}_{ji,t}$ and $\tilde{\boldsymbol{p}}_{ji,t}$ represent the true and estimated positions of A_j at time t from a local view of A_i, respectively.

We applied our context-supported relative positioning algorithm to the test dataset, and evaluated the relative position error every 2 sec. In Fig. 19, we compared the position error with a straightforward method which performs relative positioning using RSS and plain user traces without group-based correction. As a benefit of the context-supported correction mechanism, our method achieved higher positioning accuracy over most pieces of time through the experiment. The average positioning error of our method was 3.51m, which corresponds to improvement of 31.3% compared to the plain approach.

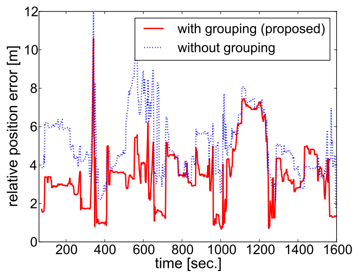

Fig. 19. Relative position error

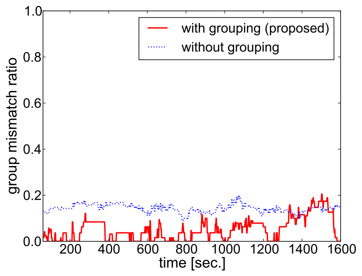

Fig. 20. Mismatch ratio between geometrical clusters and activity groups

Perceptibility of the Estimated Positions: Finally, we evaluated our system from a viewpoint of "perceptibility", which is also an important feature in PCN. As mentioned in Section 4.3, human intuitively perceives objects in close proximity as a group[14]. Based on this observation, we applied a clustering algorithm to the estimated positions to find geometrical clusters which would be regarded as a group in human perception. Comparing such geometrical clusters to actual activity groups, we evaluated the perceptibility of the estimated positions.

For geometrical clustering, we used a hierarchical clustering algorithm with the group-average method. It defines inter-cluster distance between two clusters C_1 and C_2 by the following formula:

$$d(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{p \in C_1} \sum_{q \in C_2} d(p, q) \tag{22}$$

where $d(p, q)$ is the Euclid distance between estimated positions p and q of two clients, and $|C_1|$ and $|C_2|$ represent the size of C_1 and C_2, respectively. We start the clustering from $|S|$ clusters each of which contains a different client in S. Calculating the inter-cluster distance for all the pairs of clients, we pick out a pair with the shortest distance to merge them to a single cluster. We repeat this bottom-up clustering process until the shortest distance falls to a threshold. We set this threshold to 1.5m in the evaluations below.

We evaluated the consistency between the resulting geometrical clusters and actual activity groups by failure ratio of pairwise membership test. Fig. 20 shows the result compared to the straightforward method without group-based correction. For the compared method, errors in estimated traces make the positional relationship between the clients go wrong, which leads average mismatching ratio to be 14.3%. In contrast, our method suppressed such failures to 5.1% owing to the group-based correction. This corresponds to the improvement of 64% and even be close to the average failure ratio for true positions of 3.9%.

6 Conclusions

In this paper, we proposed a novel social navigation framework, called PCN, that leads users to their friends in a crowd of neighbors. PCN provides relative positions of surrounding people based on sensor readings and Bluetooth RSS, both of which can be easily obtained via off-the-shelf mobile phones. Through a field experiment in a real trade fair, we demonstrated that PCN improves positioning accuracy by 31% compared to a conventional approach owing to its context-supported error correction mechanism. Furthermore, we showed that the geometrical clusters in the estimated positions are highly consistent with actual activity groups, which would help users to easily identify actual nearby people.

References

1. Banerjee, N., Agarwal, S., Bahl, P., Chandra, R., Wolman, A., Corner, M.: Virtual compass: relative positioning to sense mobile social interactions. In: Proc. of Pervasive 2010. pp. 1–21 (2010)
2. Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: Proc. of SIGMOD 2005. pp. 491–502 (2005)
3. Chintalapudi, K., Iyer, A.P., Padmanabhan, V.N.: Indoor localization without the pain. In: Proc. of MobiCom 2010. pp. 173–184 (2010)
4. Chitte, S., Dasgupta, S., Ding, Z.: Distance estimation from received signal strength under log-normal shadowing: Bias and variance. IEEE Signal Processing Letters 16(3), 216 –218 (2009)
5. Constandache, I., Bao, X., Azizyan, M., Choudhury, R.R.: Did you see bob?: human localization using mobile phones. In: Proc. of MobiCom 2010. pp. 149–160 (2010)
6. Constandache, I., Choudhury, R.R., Rhee, I.: Towards mobile phone localization without war-driving. In: Proc. of INFOCOM 2010. pp. 1–9 (2010)
7. Fujii, S., Nomura, T., Umedu, T., Yamaguchi, H., Higashino, T.: Real-time trajectory estimation in mobile ad hoc networks. In: Proc. of MSWiM 2009. pp. 163–172 (2009)
8. Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. In: Proc. of MobiCom 1999. pp. 59–68 (1999)
9. Higuchi, T., Fujii, S., Yamaguchi, H., Higashino, T.: An efficient localization algorithm focusing on stop-and-go behavior of mobile nodes. In: Proc. of PerCom 2011. pp. 205–212 (2011)
10. Kloch, K., Lukowicz, P., Fischer, C.: Collaborative PDR localisation with mobile phones. In: Proc. of ISWC 2011. pp. 37–40 (2011)
11. Krumm, J., Hinckley, K.: The NearMe wireless proximity server. In: Proc. of UbiComp 2004. pp. 283–300 (2004)
12. Peng, C., Shen, G., Zhang, Y., Li, Y., Tan, K.: BeepBeep: A high accuracy acoustic ranging system using COTS mobile devices. In: Proc. of SenSys 2007. pp. 1–14 (2007)
13. Steinhoff, U., Schiele, B.: Dead reckoning from the pocket — an experimental study. In: Proc. of PerCom 2010. pp. 162–170 (2010)
14. Wertheimer, M.: Laws of organization in perceptual forms (1938)
15. Woodman, O., Harle, R.: Pedestrian localisation for indoor environments. In: Proc. of UbiComp 2008. pp. 114–123 (2008)
16. Yin, J., Yang, Q., Ni, L.M.: Learning adaptive temporal radio maps for signal-strength-based location estimation. IEEE Transactions on Mobile Computing 7(7), 869–883 (2008)