Tracking Mobile Users in Wireless Networks via Semi-Supervised Colocalization

Jeffrey Junfeng Pan, *Member*, *IEEE*, Sinno Jialin Pan, *Member*, *IEEE*, Jie Yin, *Member*, *IEEE*, Lionel M. Ni, *Fellow*, *IEEE*, and Qiang Yang, *Fellow*, *IEEE*

Abstract—Recent years have witnessed the growing popularity of sensor and sensor-network technologies, supporting important practical applications. One of the fundamental issues is how to accurately locate a user with few labeled data in a wireless sensor network, where a major difficulty arises from the need to label large quantities of user location data, which in turn requires knowledge about the locations of signal transmitters or access points. To solve this problem, we have developed a novel machine learning-based approach that combines collaborative filtering with graph-based semi-supervised learning to learn both mobile users' locations and the locations of access points. Our framework exploits both labeled and unlabeled data from mobile devices and access points. In our two-phase solution, we first build a manifold-based model from a batch of labeled and unlabeled data in an offline training phase and then use a weighted k-nearest-neighbor method to localize a mobile client in an online localization phase. We extend the two-phase colocalization to an online and incremental model that can deal with labeled and unlabeled data that come sequentially and adapt to environmental changes. Finally, we embed an action model to the framework such that additional kinds of sensor signals can be utilized to further boost the performance of mobile tracking. Compared to other state-of-the-art systems, our framework has been shown to be more accurate while requiring less calibration effort in our experiments performed on three different testbeds.

Index Terms—Wireless sensor networks, semi-supervised learning, indoor localization, colocalization, AI applications.

1 Introduction

Locating users in a wireless network is an important task in many applications that range from context-aware computing [1], location-based services [2], [3] to robotics [4], [5]. With recent advances in pervasive computing and mobile technology, the problem of tracking wireless devices using received signal strength (RSS) has attracted intense interest in many research communities [6], [7]. RSS-based tracking or localization is a challenging task since radio signals usually attenuate in a highly nonlinear and uncertain way in a complex environment where client devices may be moving. Existing approaches to RSS-based localization fall into two main categories: 1) radio propagation models [8] and 2) statistical machine-learning models [9], [10], [11].

Traditionally, practitioners have used geometric models that are based on signal propagation properties and access point (AP) locations. These models have poor accuracy when the access points are separated far from each other as in cellphone base towers. More recent works have used learning-based models that can achieve much better accuracy. These learning-based models are set up purely from the client devices based on a large amount of calibration data [9], [10], [11]. However, a major problem with the learning-based models is that, in many indoor localization cases, the calibrated training data are manually collected since the Global Positioning System (GPS) may not work in an indoor environment. The data collection process is time consuming and can be easily outdated, making it necessary for us to collect the data over and over again. In order to reduce the calibration effort, this work attempts to answer the following three questions:

- How can we reduce calibration effort to build a tracking system by incorporating unlabeled data?
- Can we further enhance the performance if the locations of some access points are known?
- Can we make use of different kinds of signals to further boost the performance?

In this paper, we address the problem of simultaneously recovering the locations of both mobile devices and access points, which we call *colocalization*, using *labeled* and *unlabeled* RSS data from both mobile devices and access points. We propose two solutions to this problem. The first one is called *two-phase colocalization*, which is based on semi-supervised manifold-learning techniques, which has an offline training phase and an online localization phase. However, a two-phase model may not adapt to environmental changes well since the model remains unchanged after being trained. To solve this problem, we extend the model to *online colocalization*, which can cope with calibrated and uncalibrated data stream in real time and adjust itself online.

- **Solution I. *Two-Phase Colocalization***
  In general, learning-based systems using RSS values function in two phases : an *offline training phase* and an

- *J.J. Pan is with Facebook Inc., 1601 S. California Ave., Palo Alto, CA 94304. E-mail: panjunfeng@gmail.com.*
- *S.J. Pan is with the Institute for Infocomm Research, 1 Fusionopolis Way, # 21-01 Connexis, Singapore 138632. E-mail: sinnocat@gmail.com.*
- *J. Yin is with the Information Engineering Laboratory, CSIRO ICT Centre, Australia. E-mail: Jie.Yin@csiro.au.*
- *L.M. Ni and Q. Yang are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. E-mail: {ni, qyang}@cse.ust.hk.*

Manuscript received 19 Oct. 2010; revised 1 Apr. 2011; accepted 18 Apr. 2011; published online 4 Aug. 2011.
Recommended for acceptance by I. Reid.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2010-10-0806.
Digital Object Identifier no. 10.1109/TPAMI.2011.165.

online localization phase. In the offline phase, a *learning-based model* is trained by using the signal strength values received from the access points at selected locations in the area of interest. These values comprise the training data gathered from a physical region, which are used to calibrate a probabilistic location-estimation system. In the online localization phase, the real-time signal strength samples received from the access points are used to estimate the current location based on the learned model.

More specifically, in the offline training phase, we take two steps for model building. In the first step, we assume that only unlabeled RSS data are given. We show that the problem can be solved by Latent Semantic Indexing (LSI) or Singular Value Decomposition (SVD) [12] techniques that are popular in information retrieval. Consequently, the relative locations of access points and mobile device trajectories can be determined. In the second step, we assume that a small amount of labeled RSS data from mobile devices and access points is given. To recover the absolute locations of the devices and access points, we apply a semi-supervised algorithm with graph Laplacian and manifold learning [13], [14]. Finally, we provide a unified framework for both the above unsupervised and semi-supervised solutions. A preliminary version of this solution can be found in [15].

- **Solution II.** ***Online Colocalization***
  However, in many applications, access points cannot be deployed in a static environment where calibrated and uncalibrated data arrive in a streaming manner. Access points may be removed, relocated, and added for better coverage and link quality. In each case, a localization system may gradually become inaccurate without costly recalibration and rerunning the whole training process. It is also wasteful to discard previous computational results, even if the system can be retrained. A better idea is to construct an online localization model in a streaming manner.

  The *online colocalization* extends the *two-phase* framework and addresses the problem of recovering the locations of both mobile devices and access points from radio signals that come in a streaming manner, by exploiting both labeled and unlabeled data from mobile devices and access points. The solution is based on online and incremental manifold-learning techniques [16], [17] and semi-supervised techniques [14] that can cope with labeled and unlabeled data that come sequentially. A preliminary version of *online colocalization* can be found in [18].
- **Extension.** ***Sensor Fusion with Action Models***
  Note that the above two solutions rely on measuring signal strength values sent from static *landmarks* such as wireless access points to mobile devices. Localization systems can also be broadly classified into two categories: landmark-based and landmark-free, depending on what sensor devices are used. Landmark-based systems rely on a certain proximity measurement between a mobile device and multiple landmarks that are deployed in the environment [19], [20]. Typical landmarks can be satellites in GPS or access points in Wi-Fi networks. In an indoor environment, satellite signals are not always available. Instead, Wi-Fi access points are deployed in many buildings. However, accurately tracking mobile devices using RSS is a challenging task since RSS values have large noise in a complex indoor environment due to attenuation, shadowing, and multipath effects.

  Landmark-free systems can perform self-localization without relying on any external references [21]. For example, a mobile robot can locate itself because an *action* sequence is usually available. The robot can update its *status* after executing an *action* such as *move*(*forward, 1 meter*) or *turn*(*left, 90 degrees*), which means the robot is "*to move forward 1 meter*" or "*to turn left 90 degrees*," respectively. Similarly, an Inertial Navigation System (INS) has motion sensors such as gyroscope, accelerometer, and compass which can be used for inferring the *action* of a mobile user such as speed and orientation, walking or not, etc. Landmark-free systems can be very accurate for a short time. However, errors may be accumulated due to sensor noise if no landmarks are available for recalibration.

  Hence, a better idea is to combine the landmark-based and landmark-free systems. In this paper, we extend the proposed colocalization solutions by utilizing both signal strength received from *landmarks* and readings from motion sensors. Specifically, we use the *action* sequences inferred from compass and accelerometer, and reconstruct the location trajectory via semi-supervised manifold-learning techniques. We borrow and extend the idea from [22] in the sense that if $action_i$ and $action_j$ are similar, the change of *status* or location would be similar. Our method is called Localization via Action Respecting Manifold (LARM).

2 Related Works

In the past, propagation models were widely used for location estimation due to their simplicity and efficiency [23]. These models usually assume that access points are labeled, e.g., their locations are known. An alternative is to apply machine-learning methods to learn a model that captures the correlations between RSS values and locations [7]. With these methods, the location information of access points need not be known. Instead, they usually rely on models that are trained with RSS data collected on a mobile device and the corresponding labels or physical locations [9], [24], [11]. The training data are usually collected offline. These signal values may be noisy and nonlinear due to environmental dynamics. Therefore, sufficient data have to be collected to power algorithms for approximating the signal to location mapping functions using histograms [24], k nearest neighbors (KNN) [9], etc.

Besides semi-supervised learning models, transfer-learning techniques have been also applied to the RSS-based localization problem to reduce the calibration effort [7]. The goal of transfer learning is to learn a precise model

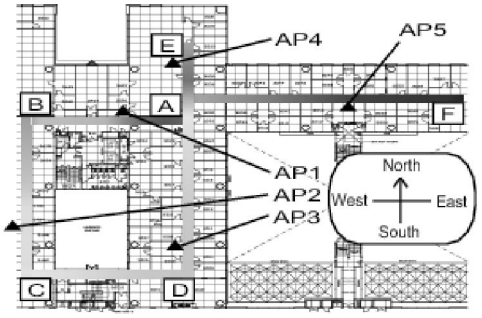


Fig. 1. An indoor WLAN testbed.

in a target domain with as few as training data by making use of training data from a related domain where the data distribution may be different from that of the target domain [25].

However, these transfer learning-based models only focused on tracking the mobile device, while our proposed colocalization framework can recover the locations of access points and track the mobile device simultaneously. By assuming an action model is given, Ferris et al. [19] proposed an unsupervised framework for simultaneous localization and mapping (SLAM) [5] in a Wi-Fi environment. It has been observed in [22] that two identical *actions* lead to similar *status* change. By treating *actions* as discrete labels, latent coordinates can be recovered via Action Respecting Embedding [22].

3 Methodology

3.1 Problem Statement

Consider a two-dimensional *colocalization* problem.[1] Assume that a user holds a mobile device and navigates in an indoor wireless environment $\mathcal{C} \subseteq \mathbb{R}^2$ with n access points which can periodically send out beacon signals. At some time t_i, the RSS values from all of the n access points are measured by the mobile device to form a row vector $\mathbf{s}_i = [s_{i_1}\ s_{i_2} \ldots s_{i_n}] \in \mathbb{R}^n$. A sequence of m signal strength vectors form an $m \times n$ matrix $S = [\mathbf{s}'_1\ \mathbf{s}'_2 \ldots \mathbf{s}'_m]'$, where $\mathbf{s}'_i$ denotes a transposition of $\mathbf{s}_i$. Here, the locations of some access points and the mobile devices at some time t_i are known or labeled, while the rest are unlabeled.

We estimate the $m \times 2$ location matrix $P = [\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_m]'$ where $\mathbf{p}_i = [p_{i_1}\ p_{i_2}] \in \mathcal{C}$ is the location of the mobile device at time t_i and the $n \times 2$ location matrix $Q = [\mathbf{q}'_1, \mathbf{q}'_2, \ldots, \mathbf{q}'_n]'$ where $\mathbf{q}_j = [q_{j_1}\ q_{j_2}] \in \mathcal{C}$ is the location of the jth access point. We call this problem *colocalization*. More specifically, we have two main objectives:

- ***Two-phase colocalization***. Given a fixed amount of labeled and unlabeled data collected offline, the first objective is to build a model for simultaneously recovering the locations of the remaining unknown access points and the trajectory of the mobile device. The model can then be used for online localization. The model remains unchanged in the online phase unless we retrain everything. These offline and online phases are done in the same way as most traditional machine-learning approaches are.
- ***Online colocalization***. Assuming that partially calibrated data come sequentially, the second objective is to determine and update the locations of the remaining unlabeled access points and the trajectory of the mobile device in real time. Note that m is not a constant value. As time elapses, m may increase from $1, 2, \ldots$ to any number. We wish to dynamically adjust the model when observing new data without relying on an offline training phase.

1. Note that it is straightforward to extend our proposed models to three-dimensional colocalization problems.

TABLE 1
Signal Strength (Unit: dBm)

	AP_1	AP_2	AP_3	AP_4	AP_5
t_A	-40		-60	-40	-70
t_B	-50	-60		-80	
t_C		-40	-70		
t_D	-80		**-40**	-70	
$t_{A'}$	-40		-70	-40	-60
t_E	-40		-70	-40	-80
t_F	-80			-80	-50

(All values are rounded for illustration)

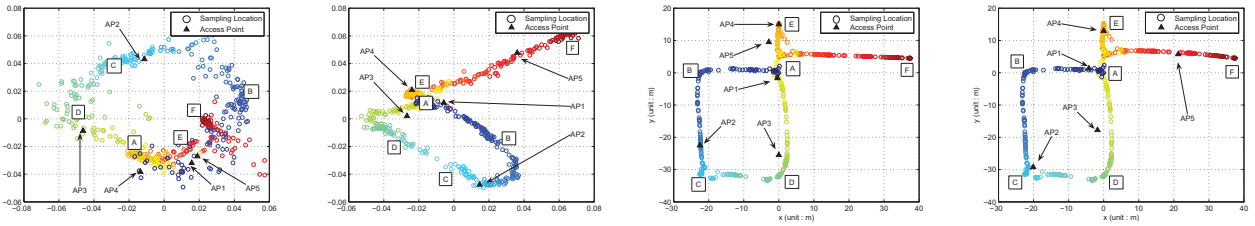
Example 1. As an example, Fig. 1 shows an indoor 802.11 wireless LAN environment of size about $60\ \text{m} \times 50\ \text{m}$, which has $n = 5$ access points. A user with an IBM T42 notebook that is equipped with an Intel Pro/2200BG internal wireless card walks from A through B, C, D, A, E to F at time $t_A, t_B, t_C, t_D, t_{A'}, t_E, t_F$. Correspondingly, a total number of $m = 1, 2, \ldots, 7$ signal strength vectors are incrementally extracted. The final 7×5 matrix S is shown in Table 1. By walking from A to F in the hallways, we collected 500 signal strength vectors from five access points. Note that the blank cells denote the missing values, which we can fill in a small default value, e.g., -100 dBm.

Our first task is to estimate the trajectory matrix P of the mobile device at all times and to determine the location matrix Q of the access points $AP_1, AP_2, \ldots, AP_5$. Our second task is to dynamically update the trajectory matrix P of the mobile device at each time when new data come and to update the location matrix Q of the access points in an online manner.

3.2 Domain Characteristics

There are four main characteristics about RSS by observing the data in Table 1:

1. Considering two *rows* of the data, the mobile device at two different times may be spatially close if the pairwise signal strengths are similar from most access points, e.g., the times t_A and $t_{A'}$.
2. Considering two *columns* of the data, two access points may be spatially close if their pairwise signal strength values are similar most of the time, e.g., AP_1 and AP_4.
3. Considering a *single cell* s_{ij} of the data, the mobile device and the j access point may be spatially close



(a) (Linear) SVD co-localization (b) (Nonlinear) SVD co-localization (c) Manifold-based co-localization (d) The unifying framework

Fig. 2. 802.11 Wireless LAN test in an indoor environment.

to each other at time t_i if the signal is strong, e.g., the mobile device is close to AP_3 at time t_D.

4. Considering two *neighbored rows* of the data, the mobile device at two consecutive times may be spatially close if their time interval is small by assuming that a user may not move too fast or too irregularly. For example, the locations of the mobile device at time $t_{A'}$ and t_E are close since $|t_{A'} - t_E| < \Delta T$.

3.3 SVD-Based Relative Colocalization

Given unlabeled data only, we can determine the relative locations of the mobile device and the access points. Not surprisingly, the relative *colocalization* is closely related to Latent Semantic Indexing [12]. In this view, we treat an access point as a term and a mobile device at some time as a document. The first three observed characteristics mentioned above would be mapped to the similarities of document-document, term-term, and document-term, respectively. Estimating the positions of the mobile device and the access points corresponds to discovering the latent semantics of documents and terms in some concept space.

More specifically, we can estimate the relative coordinates by performing Singular Value Decomposition.

1. Transform the signal matrix $S = [s_{ij}]_{m\times n}$ to a nonnegative weight matrix $\widetilde{S} = [\widetilde{s}_{ij}]_{m\times n}$ by a *linear* function $\widetilde{s}_{ij} = s_{ij} - s^{min}$, where s^{min} is the minimal signal strength detected, e.g., the noise level or -100 dBm.
2. Normalize the weight matrix by $\widetilde{S}_{\mathcal{N}} = D_1^{-1/2}\widetilde{S}D_2^{-1/2}$. Here, D_1 and D_2 are both diagonal matrices such that $D_1 = diag(d_1^1, d_2^1, \ldots, d_m^1)$ where $d_i^1 = \sum_{j=1}^{n} \widetilde{s}_{ij}$ and $D_2 = diag(d_1^2, d_2^2, \ldots, d_n^2)$ where $d_j^2 = \sum_{i=1}^{m} \widetilde{s}_{ij}$.
3. Perform SVD on the normalized weight matrix by $\widetilde{S}_{\mathcal{N}} \approx U_{m\times r}\Sigma_{r\times r}V'_{n\times r}$. The columns of $U_{m\times r} = [\mathbf{u}_1 \ldots \mathbf{u}_r]$ and $V_{n\times r} = [\mathbf{v}_1 \ldots \mathbf{v}_r]$ are the left and right singular vectors. The singular values of the diagonal matrix $\Sigma_{r\times r} = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$ are ranked in *nonincreasing* order.
4. The (latent) location matrices of the mobile device P and that of the access points Q can be estimated using $P = D_1^{-1/2}[\mathbf{u}_2\ \mathbf{u}_3]$ and $Q = D_2^{-1/2}[\mathbf{v}_2\ \mathbf{v}_3]$. Note that we skip the first singular vectors $\mathbf{u}_1$ and $\mathbf{v}_1$ which mostly capture some constant since matrix $\widetilde{S}_{\mathcal{N}}$ is not centering.

As an example, after performing SVD on data in **Example 1**, we obtained the latent coordinates of the mobile device and the access points, which are shown in Fig. 2a. In this example, it is easy to see that the hallway structure is not well preserved by comparing the true location sequence shown in Fig. 1. This is because SVD assumes a *linear* subspace, while the correlation of RSS values and distance to access points is often nonlinear [11].

A better solution is using kernelized SVD [26] by transforming signal strength values to weights by a *nonlinear* function. More specifically, we transform the signal matrix $S = [s_{ij}]_{m\times n}$ to a new weight matrix $\widetilde{S} = [\widetilde{s}_{ij}]_{m\times n}$ by a Gaussian function:

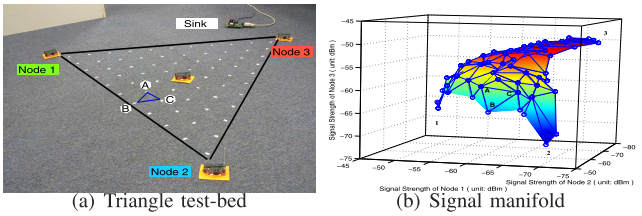
$$\widetilde{s}_{ij} = exp(-|s_{ij} - s^{max}|^2/2\sigma^2), \quad (1)$$

where s^{max} is the maximal signal strength detected, e.g., the signal strength around an access point, and σ is a parameter of the Gaussian kernel, which is known as kernel width. Fig. 2b plots the *colocalization* result using P and Q. Intuitively, the reconstructed hallway structure and the locations of access points are better than that shown in Fig. 2a while referring to the ground truth illustrated in Fig. 1.

3.4 Manifold-Based Absolute Colocalization

When the physical locations of some access points and the mobile device at some time are known, we can ground the unknown coordinates by exploiting the geometry of the signal distribution. More specifically, we can use manifold-based learning, which generally assumes that if two points are close in the intrinsic geometry of the marginal distribution, their conditional distributions are similar [27]. This implies that mobile devices would be spatially close to each other if their signal vectors are similar along some manifold structure [28]. For example, the mobile device at times t_A and t_E would be spatially close to each other (Fig. 1) since their signal strength values are similar (Table 1).

A more concrete example is shown in Fig. 3. As can be seen in Fig. 3a, there is a two-dimensional triangle localization area with three beacon nodes placed at the vertices. The corresponding signal strength values form a two-dimensional nonlinear signal manifold in a three-dimensional space in Fig. 3b. Points A, B, and C are neighbors in both location and signal spaces.



(a) Triangle test-bed (b) Signal manifold

Fig. 3. Neighborhood preserving.

When the manifold assumption holds, the optimal solution is given by $\mathbf{f}^* = \arg\min \Sigma_{i=1}^{l} |f_i - y_i|^2 + \gamma \mathbf{f}^T L \mathbf{f}$ [14], where the first term measures the fitting error and the second term poses the smoothness along the manifold and L is the graph Laplacian [29]. For our problem, the objective is to optimize:

$$P^* = \arg\min_{P \in \mathbb{R}^{m \times 2}} (P - Y_P)' J_P (P - Y_P) + \gamma_P P' L_P P, \quad (2)$$

where P is the coordinate matrix of the mobile device to be determined, $J_P = diag(\delta_1, \delta_2, \ldots, \delta_m)$ is an indication matrix where $\delta_i = 1$ if the coordinate of the mobile device at time t_i is given and otherwise $\delta_i = 0$, $Y_P = [\mathbf{y}'_1, \mathbf{y}'_2, \ldots, \mathbf{y}'_m]'$ is an $m \times 2$ matrix supplying the calibration data where $\mathbf{y}_i$ is the given coordinate of the mobile device at time t_i if $\delta_i = 1$ and otherwise the value of $\mathbf{y}_i$ can be any, e.g., $\mathbf{y}_i = [0\ 0]$, γ_P controls the smoothness of the coordinates along the manifold, $L_P = D_P - W_P$ is the graph Laplacian, $W_P = [w_{ij}]_{m \times m}$ is the weight matrix and $w_{ij} = exp(-\|\mathbf{s}_i - \mathbf{s}_j\|^2 / 2\sigma^2)$ if $\mathbf{s}_i$ and $\mathbf{s}_j$ are neighbors along the manifold and otherwise $w_{ij} = 0$, $D_P = diag(d_1, d_2, \ldots, d_m)$ and $d_i = \sum_{j=1}^{m} w_{ij}$.

By setting the derivative of the right-hand side in (2) to zero, we obtain the optimal solution shown as follows:

$$P^* = (J_P + \gamma_P L_P)^{-1} J_P Y_P. \quad (3)$$

Similarly, the coordinates of the access points can be obtained by solving the following optimization problem $Q^* = \arg\min_{Q \in \mathbb{R}^{n \times 2}} (Q - Y_Q)' J_Q (Q - Y_Q) + \gamma_Q Q' L_Q Q$, and thus

$$Q^* = (J_Q + \gamma_Q L_Q)^{-1} J_Q Y_Q, \quad (4)$$

where $L_Q = D_Q - W_Q$ is the graph Laplacian, W_Q is the weight matrix, and D_Q is constructed from W_Q.

Thus, when the locations of the mobile device and the access points are partially known, we can *colocalize* them by solving (3) and (4), respectively. Alternatively, we can combine them into a single equation as

$$R^* = (J + \gamma_B L_B + \gamma_C L_C)^{-1} JY, \quad (5)$$

where $R = [P'\ Q']'$ is the coordinate matrix of the mobile device and the access points, $Y = [Y'_P\ Y'_Q]'$ gives the label information, $J = \left[\begin{smallmatrix} J_P & 0 \\ 0 & J_Q \end{smallmatrix}\right]$ is the indication matrix, and $L_B = \left[\begin{smallmatrix} L_P & 0 \\ 0 & 0 \end{smallmatrix}\right]$ and $L_C = \left[\begin{smallmatrix} 0 & 0 \\ 0 & L_Q \end{smallmatrix}\right]$ are the graph Laplacians.

In practice, the graph Laplacians L_B and L_C in (5) are normalized [13], [30]. Fig. 2c shows an example of the manifold-based *colocalization* when the locations of the mobile device at times t_A, t_B, t_C, t_D, t_E, and t_F and the access points AP_2, AP_3, and AP_4 are known. As can be seen, the trajectory of the mobile device is well grounded when compared to the ground truth shown in Fig. 1. However, locations of access points are estimated badly, e.g., the location of AP_5. The reason is that in manifold-based colocalization, there are two manifolds, one is for Wi-Fi data and the other is for access points. These two manifolds are learned separately. Most manifold-based methods require dense unlabeled data to propagate label information through an underlying manifold structure. However, from the access points' perspective, the data are extremely sparse. Furthermore, AP_5 is far away from the other four. In this case, the manifold-based colocalization approach is not able to estimate the location of AP_5 accurately. In contrast, the SVD-based colocalization approach employs matrix factorization techniques to recover *latent* locations of the access points and Wi-Fi data jointly. As a result, a lot of unlabeled Wi-Fi data can help recover latent locations of the access points. Although the latent coordinates cannot be aligned to absolute locations without label information, the relative distance between access points is more accurate than that estimated by the manifold-based colocalization approach. In the following, we propose to combine SVD-based and manifold-based *colocalization* to align the mobile device and the access points to the ground truth jointly.

3.5 Solution I: Two-Phase Colocalization

Offline training phase. So far, we have formulated the unsupervised *colocalization* based on SVD and the semi-supervised *colocalization* based on the manifold assumption using (5) by exploiting the correlation between the mobile device and the access points. In this section, we integrate them through a unifying framework. Essentially, performing SVD on $S_{\mathcal{N}}$ is equivalent to solving the following generalized eigenvalue problem [31]:

$$L_A Z = D_A Z \Lambda, \quad (6)$$

where $L_A = D_A - W_A$ is the graph Laplacian, $W_A = \left[\begin{smallmatrix} 0 & \widetilde{S}' \\ \widetilde{S} & 0 \end{smallmatrix}\right]$, and $D_A = \left[\begin{smallmatrix} D_1 & 0 \\ 0 & D_2 \end{smallmatrix}\right]$. The eigenvalues of the diagonal matrix $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_{m+n})$ are ranked in *nondecreasing* order. $Z = [\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_{m+n}]$ are the eigenvectors. $[P'\ Q']' = [\mathbf{z}_2\ \mathbf{z}_3]$. Note that we skip the first eigenvector $\mathbf{z}_1$ since the solution is trivial. Furthermore, it is interesting to see that we have $\lambda_i = 1 - \sigma_i$, where $i = 1, 2, \ldots, r$ [31]. Detailed analysis and comparison of LSI, SVD, and graph Laplacian can be found in the literature on LSI [12], Bipartite Co-Clustering [31], and Fiedler Embeddings [32].

Putting (5) and (6) together, we aim to optimize:

$$R^* = \arg\min_{R \in \mathbb{R}^{(m+n) \times 2}} (R - Y)' J (R - Y) + \gamma R' L R. \quad (7)$$

The first term measures the fitting error and the second term constrains the smoothness among the mobile device and the access points. The solution is given by

$$R^* = (J + \gamma L)^{-1} JY, \quad (8)$$

where $L = \gamma_A L_A + \gamma_B L_B + \gamma_C L_C = D - W$.

We set γ to a small positive value, which is directly related to harmonic functions on the graph such that the coordinate of a mobile device or an access point $\mathbf{r}_i$ in $R = [\mathbf{r}'_1, \mathbf{r}'_2, \ldots, \mathbf{r}'_{m+n}]'$ is determined by the average of its neighbors:

$$\mathbf{r}_i = \frac{\sum_j w_{ij} \mathbf{r}_j}{\sum_j w_{ij}},$$

where

$$W = [w_{ij}]_{(m+n) \times (m+n)} = \begin{bmatrix} \gamma_B W_P & \gamma_A \widetilde{S} \\ \gamma_A \widetilde{S}' & \gamma_C W_Q \end{bmatrix}.$$

In practice, we optimize the objective function over the *normalized* graph Laplacian [13], [30] to balance the weights of vertices by substituting $R = D^{-1/2} F$ into (7):

$$F^* = \arg\min_{F \in \mathbb{R}^{(m+n)\times 2}} (D^{-1/2}F - Y)' J (D^{-1/2}F - Y) + \gamma_{\mathcal{N}} F' L_{\mathcal{N}} F, \tag{9}$$

where $L_{\mathcal{N}} = D^{-1/2} L D^{-1/2}$ is the *normalized* graph Laplacian. The optimal F is given by

$$F^* = (JD^{-1/2} + \gamma_{\mathcal{N}} L_{\mathcal{N}})^{-1} JY. \tag{10}$$

Substituting $F = D^{1/2}R$ back into (10), the locations of the mobile device and the access points are given by

$$R^* = D^{-1/2}(JD^{-1/2} + \gamma_{\mathcal{N}} L_{\mathcal{N}})^{-1} JY. \tag{11}$$

We can export the estimated coordinates of the mobile device trajectory P^* and the access point locations Q^* from $R^* = [P^{*\prime}\ Q^{*\prime}]'$.

Online localization phase. The location of a new signal strength vector $\mathbf{s}_i$ is predicted as follows:

1. Find the k neighbors closest to $\mathbf{s}_i$ in the training data $S = [\mathbf{s}'_1\ \mathbf{s}'_2 \dots \mathbf{s}'_m]'$. Let $\mathcal{C}_i$ be the index set of the k nearest neighbors. Besides, we link $\mathbf{s}_i$ to those access points from which we can detect the radio signal. We also link $\mathbf{s}_i$ to $\mathbf{s}_{i-1}$ in order to pose the temporal constraint by assuming that a user may not move too fast ($t_i - t_{i-1} < \Delta T$). Denote the index set for these additional links as $\mathcal{B}_i$.
2. Approximately, we can predict the location using harmonic functions [33], which are smooth functions on the graph such that $\mathbf{r}_i$ is determined by the weighted average of its neighbors. This property holds if there is no uncertainty in the labeled locations of matrix P during training ($\gamma \to 0$ in (7)):

$$\tilde{\mathbf{r}}_i \approx \frac{\sum_{j \in \mathcal{C}_i \cup \mathcal{B}_i} w_{ij} \mathbf{r}_j}{\sum_{j \in \mathcal{C}_i \cup \mathcal{B}_i} w_{ij}}. \tag{12}$$

Note that the above $\tilde{\mathbf{r}}_i$ is an approximation because adding $\mathbf{s}_i$ to the existing neighborhood graph from the training data may slightly change the graph structure. We link the ith node to the node set $\mathcal{C}_i$ but do not eliminate any existing edge in the graph to maintain the k-neighbor relationship among all nodes.

3.6 Solution II: *Online Colocalization*

We will extend the above *Two-Phase Colocalization* model to an *online* version. We wish that it can dynamically adjust itself when new data come sequentially in real time. The key point is how to add the new data into the learned graph by updating the k-neighbor relationship and the corresponding weight matrix W. This can be done repeatedly in two online steps: Predict and Update.

Predict. Given a new signal vector $\mathbf{s}_i$ at time t_i, we find its k nearest neighbors and use (12) in the above *online localization phase* for predicting the location $\tilde{\mathbf{r}}_i$.

Update. The addition and deletion of nodes can modify the neighborhood graph and the corresponding graph Laplacian. We use the method described in [16] for updating the neighborhood graph structure locally.

- **Node addition.** Let $\mathcal{A}_i^+$ and $\mathcal{D}_i^+$ be the set of edges to be added and deleted after inserting v_i to the neighborhood graph, respectively. Let τ_j be the index of the kth nearest neighbor of v_j. Here, we assume that all k nearest neighbors of v_j have been ranked in *nondecreasing* order in terms of the distance to v_j. Given a k-nearest-neighborhood graph consisting of n nodes, when the $(n+1)$th node v_i is inserted into the graph, we need to add k edges to connect v_i to its k nearest neighbors, $e(v_i, v_j)$, where $v_j \in \mathcal{C}_i$. Furthermore, for each v_j in the old graph, if $\Delta_{j,\tau_j} \le \Delta_{j,i}$, where $\Delta_{j,i}$ denotes the distance between v_i and v_j, then the k nearest neighborhood of v_j remains the same; thus, the corresponding local neighborhood graph of v_j does not need to be updated. Otherwise, if $\Delta_{j,\tau_j} > \Delta_{j,i}$, then v_i replaces v_{τ_j} in the k nearest neighborhood of v_j. Thus, the corresponding neighborhood graph needs to be updated as follows:

$$\mathcal{A}_i^+ = \{e(j,i)\ :\ j \in \mathcal{C}_i \text{ or } \Delta_{j,\tau_j} > \Delta_{j,i}\},$$
$$\mathcal{D}_i^+ = \{e(j,\tau_j)\ :\ \Delta_{j,\tau_j} > \Delta_{j,i}\ \&\ \Delta_{\tau_j,j} > \Delta_{\tau_j,l_j}\},$$

where l_j is the index of the kth nearest neighbor of v_{τ_j} after inserting v_i in the graph.
- **Node deletion.** Similarly, let $\mathcal{A}_i^-$ and $\mathcal{D}_i^-$ denote the set of edges to be added and deleted after removing v_i from the neighborhood graph, respectively. The graph update can be done as follows:

$$\mathcal{A}_i^- = \{e(i,h_i)\},\ \text{where } h_i \text{ is the } (k+1)\text{th nearest neighbor before removing } v_i \text{ in the graph.}$$
$$\mathcal{D}_i^- = \{e(i,j)\ :\ j \in \mathcal{C}_i\}.$$

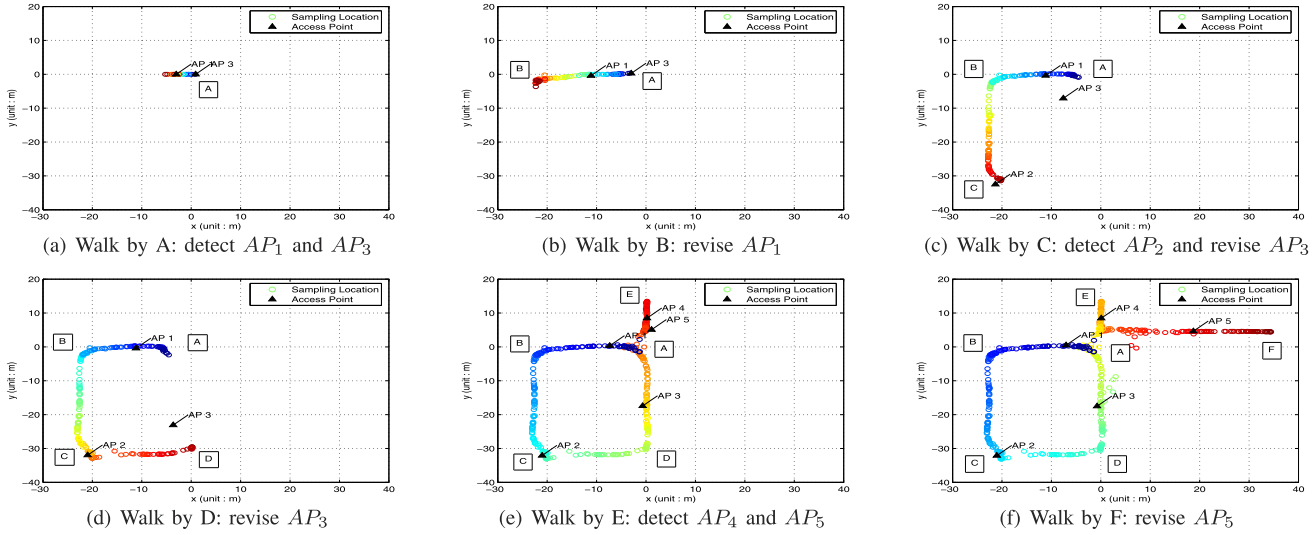
After updating the neighborhood graph, it is straightforward to modify the corresponding weight matrix W. For an added edge $e(i,j)$, we set both the values of w_{ij} and w_{ji} because the neighborhood graph is symmetric. If it is a deleted edge, we clear the values of w_{ij} and w_{ji}. The graph Laplacian $L = D - W$ can be updated in a similar way.

Finally, we have to reestimate the location matrix $R = [P'\ Q']'$ of the mobile devices and the access points so that it can reflect the change of the neighborhood graph and the new graph Laplacian L. Instead of using (8) for solving R, we update R by iteration. In each iteration cycle, we apply

$$\mathbf{r}_i^{new} = \frac{\sum_{j \in \mathcal{C}_i \cup \mathcal{B}_i} w_{ij} \mathbf{r}_j^{old}}{\sum_{j \in \mathcal{C}_i \cup \mathcal{B}_i} w_{ij}}, \quad i = 1, \dots, m+n.$$

We use the predicted $\tilde{\mathbf{r}}_i$ as the initial values for iteration. Furthermore, the weight matrix W does vary too much after addition or deletion. We can thus obtain very good estimation after a few iterations.

Example 2. A user with a mobile device walks in the office area shown in Fig. 1. The mobile device periodically collects signal vectors. The user can mark down his location when he walks by some landmark points such as corners and dead ends of the hallways $(A, B, \dots, F)$. Thus, the data that come in a streaming manner are partially labeled. By applying the *online colocalization* method, we continuously update the recovered locations of the mobile devices and the access points. Fig. 4 shows



(a) Walk by A: detect AP_1 and AP_3 (b) Walk by B: revise AP_1 (c) Walk by C: detect AP_2 and revise AP_3

(d) Walk by D: revise AP_3 (e) Walk by E: detect AP_4 and AP_5 (f) Walk by F: revise AP_5

Fig. 4. Illustration of the online colocalization when a user walks from A through B, C, D, E to F.

the *online colocalizaiton* results at six key frames when the user walks by $A, B, \ldots, F$. As can be seen, the locations of the user trajectory and the access points are dynamically calibrated when obtaining new data. For example, AP_3 gradually converges to its true location.

3.7 Special Cases of Colocalization

Colocalization is a general framework for RSS-based tracking and mapping. It addresses the problem of simultaneously recovering the locations of both mobile devices and access points by exploiting both labeled and unlabeled data from mobile devices and access points. The model can be applied with or without an offline training phase. It is flexible since we can calibrate the system in many different ways, depending on what information we have at hand. For example, if a wireless provider is unable to provide us with some access point locations, we can still set up an accurate tracking system by collecting data ourselves. If the access point locations are partially known, we can use them and further enhance the performance. Some special cases of our model are summarized as follows:

- When only unlabeled RSS data collected by mobile devices and no location information of access points are available, we can do unsupervised dimension reduction and recover the relative coordinates of both access points and mobile devices as shown in Fig. 2b. It is related to a Gaussian Process Latent Variable Model to recover latent coordinates of user trajectories based on unlabeled data [19].
- When labeled RSS data collected by mobile devices and no location information of access points are available, the model acts similarly to a classical KNN-based method [9], which is applied for indoor tracking using Wi-Fi signal strength values.
- When partially labeled RSS data collected mobile devices and no location information of access points are available, the model performs similarly to *LeMan* [28], which is a semi-supervised algorithm for sensor-network-based localization based on manifold learning. *LeMan* calibrates a tracking system purely from the client site.
- In general, when RSS data collected mobile devices and locations of access points are partially labeled, we can use all the available data for model building and get a better result than using part of the information only. We have studied how the labeled and unlabeled data help colocalization in [15], [18].

4 Extension with Action Models

As we describe in Section 1, localization systems can be classified into two categories: landmark-based and landmark-free. Landmark-based systems rely on the measurement between a tracking target and multiple landmarks such as the received signal strength between a Wi-Fi client and multiple access points. Landmark-free systems can localize themselves without the need of external references. An Inertial Navigation System can continuously update its position from measured velocity and time. Sensor readings may be inaccurate and noisy in either category of systems. Wi-Fi signal has large noise in a complex indoor environment due to shadowing and multipath effects. Inertial systems produce inaccurate dead reckoning over long periods, but accurately estimate relative motion over short intervals. In this section, we leverage the use of multiple sensors and extend the localization framework as learning Action Respecting Manifold.

4.1 Problem Restatement

Similarly to the problem statement described in Section 3.1, assume that a user holds a mobile device and navigates in a two-dimensional indoor wireless environment $\mathcal{C} \subseteq \mathbb{R}^2$ with n access points which can periodically send out beacon signals. At some time t_i, the RSS values from all the n access points are measured by the mobile device to form a row vector $\mathbf{s}_i = [s_{i1}\ s_{i2}\ \ldots\ s_{in}] \in \mathbb{R}^n$. A sequence of m signal strength vectors forms an $m \times n$ matrix $S = [\mathbf{s}'_1\ \mathbf{s}'_2\ \ldots\ \mathbf{s}'_m]'$. Furthermore, the mobile device has additional sensors for measuring the activity of the mobile user. Such sensors can be compass or accelerometer, from which we can estimate

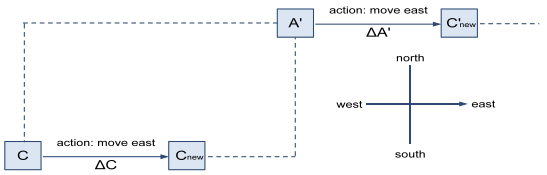


Fig. 5. Similar actions (move east) result in similar change of location (ΔC and $\Delta A'$).

the moving direction and speed. Let the speed at time t_i be o_i and the direction or azimuth be θ_i. In this paper, azimuth is measured in angle in degrees. It ranges in $[0°, 360°)$ and $0° = North$, $90° = East$, $180° = South$, $270° = West$. We denote $O = [o_1\ o_2\ \ldots\ o_m]'$ and $\Theta = [\theta_1\ \theta_2\ \ldots\ \theta_m]'$ column vectors of the sequences of speed and azimuth, respectively.

The locations of the mobile device at some time t are labeled, while the rest are unlabeled. Furthermore, locations of some access points are known, while the rest are unknown. Our objective is to estimate the $m \times 2$ location matrix $P = [\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_m]'$ and $n \times 2$ location matrix $Q = [\mathbf{q}'_1, \mathbf{q}'_2, \ldots, \mathbf{q}'_n]'$, where $\mathbf{p}_i = [p_{i_1}\ p_{i_2}] \in \mathcal{C}$ and $\mathbf{q}_j = [q_{j_1}\ q_{j_2}] \in \mathcal{C}$ are the location of the mobile device at t_i and the location of the jth access point, respectively.

Example 3. Again, Fig. 1 shows an indoor Wi-Fi environment with five access points deployed. A user holds a mobile device and walks from A through $B, \ldots, E$ and finally stops at F at time $t_A, t_B, \ldots, t_F$. Besides the signal strength vectors collected in Table 1, we can get azimuth vectors from the compass sensor, e.g., $\Theta = [270°\ 180°\ 90°\ 0°\ 180°\ 90°\ 90°]'$, and estimate the walking speed from the accelerometer sensor. Assuming the user walks at a constant speed (1 m/s) and stops at F, the speed vector is estimated as $O = [1\ 1\ 1\ 1\ 1\ 1\ 0]'$. Our task is to estimate the trajectory matrix P of the mobile device and the location matrix Q of the access points.

4.2 Signal Characteristics

Besides all the domain characteristics described in Section 3.2, there is one more important feature that explores the connection between *actions* and location changes:

- Consider two *actions* inferred from the motion sensors. If their actions are similar, the location change may also be similar. For example, in Fig. 1, a user walks from A through B, C, D, A, E to F at times $t_A, t_B, t_C, t_D, t_{A'}, t_E, t_F$. The *actions* of the mobile device are both $move(east)$ at times t_C and $t_{A'}$; the change of their locations ΔC and $\Delta A'$ should be similar. Note that the change of locations is a vector, having distance and direction of changes (Fig. 5).

4.3 Dead Reckoning Localization

Let the initial position, speed, and azimuth of the mobile device be $\mathbf{p}_1$, o_1, and θ_1. We set up a coordinate system using $\mathbf{p}_1$ as the origin, east as the positive x-axis and north as the positive y-axis. The location can be updated with $\mathbf{p}_{i+1} = \mathbf{p}_i + \Delta\mathbf{p}_i\ \ (i = 1, \ldots, m-1)$, where $\mathbf{p}_{i+1}$ and $\mathbf{p}_i$ are the locations at times t_{i+1} and t_i. $\Delta\mathbf{p}_i$ is the displacement in the interval $\Delta t_i = t_{i+1} - t_i$. More specifically,

$$\Delta\mathbf{p}_i = [\Delta p_{i_1}\ \Delta p_{i_2}] = \begin{bmatrix} o_i * \Delta t_i * sin(\theta_i) \\ o_i * \Delta t_i * cos(\theta_i) \end{bmatrix}', \tag{13}$$

where o_i and θ_i can be inferred from accelerometer and compass sensors, respectively.

Alternatively, we can reformulate it as an optimization problem. The objective is to minimize $\sum_{i=1}^{m-1}((\mathbf{p}_{i+1} - \mathbf{p}_i) - \Delta\mathbf{p}_i)^2$. Again, we can rewrite it as a matrix form:

$$P^* = \arg\min_{P \in \mathbb{R}^{m\times 2}} (G_P P - \Delta P)' J_{\Delta P} (G_P P - \Delta P), \tag{14}$$

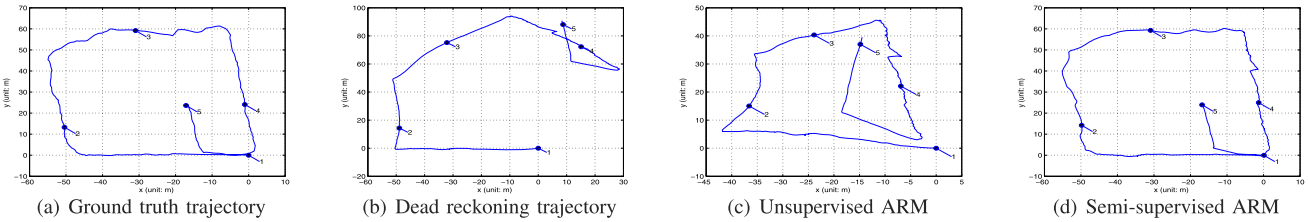
where $G_P = (g_{ij})_{m\times m}$. If $1 \le i \le m-1\ \&\ i = j$, then $g_{ij} = 1$, else if $1 \le i \le m-1\ \&\ i = j-1$, $g_{ij} = -1$, otherwise $g_{ij} = 0$. P is the coordinate matrix of the mobile device to be determined, and $J_{\Delta P} = diag(\delta_1, \delta_2, \ldots, \delta_{m-1}, 0)$ is an indication matrix where $\delta_i = 1$ if the *action* information of the mobile user at time t_i is available and otherwise $\delta_i = 0$, and $\Delta P = [\Delta\mathbf{p}'_1\ \Delta\mathbf{p}'_2\ \ldots\ \Delta\mathbf{p}'_{m-1}\ \mathbf{0}']'$ is an $m \times 2$ matrix supplying the calibration data where $\Delta\mathbf{p}_i$ is the change of location from time t_i to t_{i+1} if $\delta_i = 1$ and otherwise the value of $\Delta\mathbf{p}_i$ can be any. By setting the derivative of the right-hand side in the optimization problem (14) to zero, we can get a close form solution $P = (G'_P J_{\Delta P} G_P)^{-1} G'_P J_{\Delta P} \Delta P$.

Note that matrix G_P is singular, and thus matrix inverse is not applicable. One may consider using pseudo-inverses. Alternatively, we can pose a regularization term $P'P$ to the objective function as many machine-learning methods do. Meanwhile, we borrow the idea from Action Respecting Embedding [22] and add another term $P'G'_P L_{\Delta P} G_P P$ to measure the smoothness of *actions*. Then, we get a new optimization problem as

$$\begin{aligned} P^* = \arg\min_{P \in \mathbb{R}^{m\times 2}} \quad & \alpha(G_P P - \Delta P)' J_{\Delta P} (G_P P - \Delta P) \\ & + \beta P' G'_P L_{\Delta P} G_P P + \epsilon P' P, \end{aligned} \tag{15}$$

where α, β, and ϵ are parameters to balance the loss function term, the smoothness of *actions*, and the "complexity" of P, respectively. $L_{\Delta P}$ is the Graph Laplacian for describing the similarity of *action* pairs. Again, the similarity is described with Gaussian Kernel: $w_{ij} = exp(-\|\Delta\mathbf{p}_i - \Delta\mathbf{p}_j\|^2 / 2\sigma^2_{\Delta P})$. By setting the derivative of the right-hand side in (15) to zero, we can get a close form solution.

Example 4. Fig. 6a shows that a user holds a mobile device and walks in an area of 70 m × 80 m from point $1, 2, \ldots,$ to 5. The device can measure the Wi-Fi signal strength from the surrounding access points periodically. Meanwhile, the mobile device has digital compass and accelerometer sensors mounted so that we can estimate a sequence of azimuth θ_i and speed o_i, which will be converted to $\Delta\mathbf{p}_i$ using (13). The localization result is obtained by solving (15). Fig. 6b illustrates the localization trajectory. Compared to the ground-truth trajectory shown in Fig. 6a, the estimated locations are accurate at an initial stage, say from point 1 to 2. It gradually becomes inaccurate because the error is accumulated as time elapses. Note that we use an uncalibrated compass for collecting data. Therefore, the azimuth reading may not be accurate. However, a calibrated compass may be disturbed and become inaccurate if it is close to some local magnetic fields such as elevators.



(a) Ground truth trajectory (b) Dead reckoning trajectory (c) Unsupervised ARM (d) Semi-supervised ARM

Fig. 6. Localization comparison with motion sensors.

4.4 Extension: The LARM Algorithm

By combing the dead reckoning objective (15) and the manifold-based objective (2) together, we optimize

$$P^* = \arg\min_{P \in \mathbb{R}^{m \times 2}} (P - Y_P)' J_P (P - Y_P) + \beta P' G_P' L_{\Delta P} G_P P + \epsilon P' P + \gamma_P P' L_P P + \alpha (G_P P - \Delta P)' J_{\Delta P} (G_P P - \Delta P).$$

The first term is the fitting error to labeled data. The second term describes the smoothness on the *action* manifold. The third term poses a penalty on the complexity of the solution. The fourth term is the smoothness on the signal manifold. The fifth term is the agreement of location changes.

Example 5. By combining all sensor readings together, we can recover a better location map for either unsupervised or semi-supervised cases. Fig. 6c shows the localization result without any labeled location. If we compare Fig. 6c to Fig. 6b, we can see that unsupervised LARM can greatly correct the drifting error of Dead Reckoning by using additional unlabeled Wi-Fi signal data. Additionally, if we have 5 percent of random labeled locations, the recovered trajectory, as shown in Fig. 6d, will be pretty close to the ground truth.

Note that a motion or action model is unavailable for access points. Thus, we can combine (15) with (7) or (9) and form the objective of action-guided colocalization. The essential idea is to incorporate possible constraints about the similarity among signals and locations of mobile devices and access points, location changes and actions, etc. The new optimization problem of *action-guided colocalization* can be written as follows:

$$R^* = \arg\min_{R \in \mathbb{R}^{(m+n) \times 2}} (R - Y)' J (R - Y) + \gamma R' L R + \beta R' G' L_{\Delta R} G R + \alpha (G R - \Delta R)' J_{\Delta R} (G R - \Delta R) + \epsilon R' R,$$

where $R = [P'\ Q']'$ is the coordinate matrix of the mobile device and the access points, $Y = [Y_P'\ Y_Q']'$ gives the label information, and $J = \left[\begin{smallmatrix} J_P & 0 \\ 0 & J_Q \end{smallmatrix}\right]$ is the indication matrix. $L = \gamma_A L_A + \gamma_B L_B + \gamma_C L_C$, which is the same as defined in Section 3.5 $G = [G_P'\ \mathbf{0}']'$, $L_{\Delta R} = [L_{\Delta P}'\ \mathbf{0}']'$, $\Delta R = [\Delta P'\ \mathbf{0}']'$, and $J_{\Delta R} = [J_{\Delta P}'\ \mathbf{0}']'$. Similarly, a close form solution of the optimization problem can be obtained by

$$R = (J + \gamma L + \alpha G' J_{\Delta R} G + \beta G' L_{\Delta R} G + \epsilon I)^{-1} \times (JY + \alpha G' J_{\Delta R} \Delta R). \quad (16)$$

Note that the above solution still works even if all locations are unlabeled and *actions* are partially labeled.

4.5 Action Recognition

In previous sections, we assumed the walking speed and direction can be estimated from accelerometer and compass sensors. While it is straightforward to obtain direction readings $\Theta = [\theta_1, \theta_2, \ldots, \theta_m]'$ from compass sensors, we have not yet described any detail on how to estimate the speed $O = [o_1, o_2, \ldots, o_m]'$. Whether a user is running, walking, or standing still can be inferred from accelerometer sensors. When we recognize that the user is running or walking, the speed can be estimated via step counting, assuming the step size is known and fixed.

To recognize the user actions, we transform the sequential accelerometer data into a fixed dimension of features. More specifically, we apply a sliding window on the signals and extract the mean value, standard deviation, Cepstrum (a feature widely used in speech recognition) on each dimension of readings and Pearson correlation between pairs of dimensions. More features and models can be found in many previous works [34]. We collect a set of data and train a Support Vector Machine with Linear Kernel for action recognition. Table 2 shows the experimental results for six different actions from a different set of accelerometer data. The overall accuracy is 88.55 percent. Note that the *action* "turn left" or "turn right" may be ambiguous, while the *action* "static" is easy to be recognized. In practice, we do not need to recognize the actions on "turning" since the compass sensor has richer information.

We further count the steps once we recognize that the user is walking or climbing upstairs/downstairs. To properly segment steps, we implement some cycle detection algorithm. We apply Fast Fourier Transformation (FFT) to detect whether there is a cycle in frequency domain. FFT is an efficient method for transforming signals from time domain to frequency domain [35]. By applying FFT, we can detect a strong cycle signal in the frequency domain based on periodic patterns. Assuming that the user has a fixed step size, we can estimate the walking speed reasonably well.

TABLE 2
Action Recognition Using Accelerometer Sensor Only

estimation / truth	static	upstairs	downstairs	walk	left turn	right turn
static	994	0	0	1	7	6
upstairs	0	959	102	115	42	5
downstairs	0	36	967	56	34	9
walk	0	1	0	4457	112	131
left turn	0	50	1	117	948	15
right turn	0	0	5	248	24	314

TABLE 3
The Experimental Setups of WLAN, WSN, and RFID

	AP	MD	test-bed	scale	motion pattern
WLAN	25 APs	1 notebook	hallway	$60 \times 50m^2$	mobile (human)
WSN	8 nodes	1 mobile node	room	$5 \times 4m^2$	mobile (robot)
RFID	4 readers	30 RFID tags	room	$5 \times 4m^2$	static

5 Experimental Setup

In this section, we first evaluate the performance of the *colocalization* algorithms on three sets of different devices and testbeds. They are wireless local area networks (WLAN), wireless sensor networks (WSN), and radio frequency identification networks (RFID). In the past, researchers have tried to formalize various metrics for evaluating activity recognition and location-based services [36]. A summary of our three experimental setups is shown in Table 3.

A person carrying an IBM T42 notebook, which is equipped with an Intel Pro/2200GB internal wireless card, walks in an indoor environment of about $60\ \mathrm{m} \times 50\ \mathrm{m}$ in size. An IEEE 802.11b wireless network in the 2.4 GHz frequency bandwidth has been set up in the indoor environment. We can detect more than 20 access points. The person walks in the hallways and a total of 2,000 examples (vectors of RSS values) are collected with sample rate 2 Hz. The ground-truth location labels are obtained by referring to landmark points such as doors, corners, and dead ends. The localization area is composed of one-dimensional hallways.

The sensor-based tracking experiment was performed in the Pervasive Computing Laboratory at the Hong Kong University of Science and Technology. The room was set up as an experimental testbed of 5.0 m by 4.0 m.

We use CrossBow MICA2 and MICA2Dot to construct a wireless sensor network. We program these sensor nodes to broadcast and detect beacon frames periodically so that they can measure the RSS from each other. By combining the RSS from different nodes, we can estimate locations of these nodes. We configure all the nodes such that each of them can measure the RSS from the remaining eight nodes every 0.5 s. We tried different kinds of robots that could run freely around the floor such as Sony AIBO dogs, LEGO Mindstorms, and off-the-shelf toy cars. A *Camera Array* was used to record experiments for supporting location information (ground truth) of mobile robots. Each camera monitored at least one-fourth of the testbed. The central area was covered by all four cameras. We used some landmarks to do camera calibration such as static sensor nodes for which locations are known.

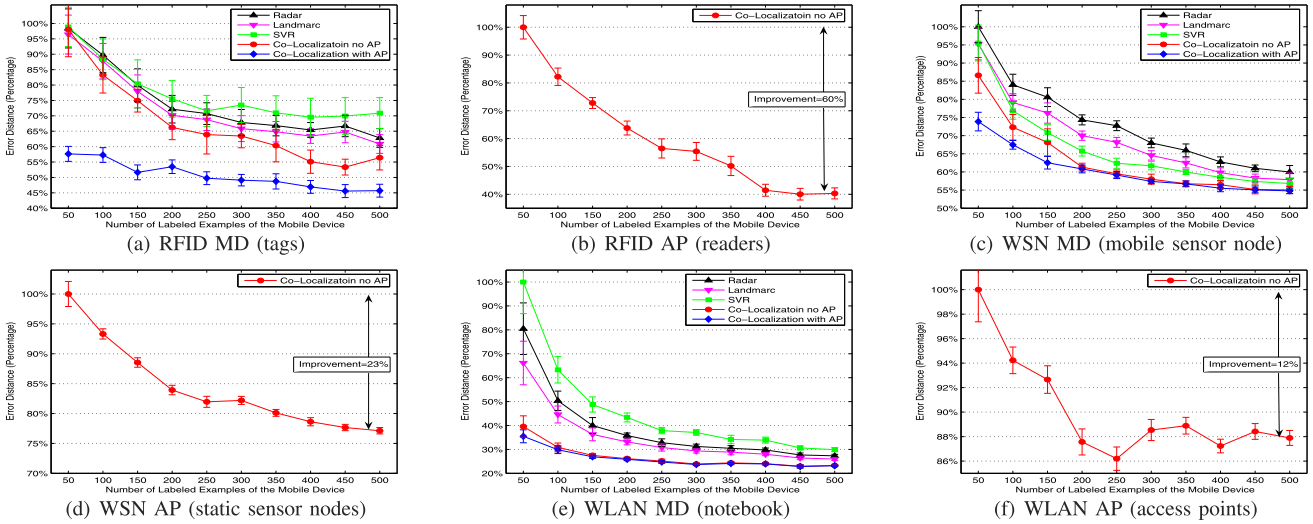
For the RFID experiment, we used four Mantis readers (AP) and 30 tags (MD) from RF Code. They were all deployed as stationary nodes. All the tags were deployed at 6×5 grid points on the $5.0\ \mathrm{m} \times 4.0\ \mathrm{m}$ floor. A total of 2,000 examples with ground-truth locations were collected.

6 Experimental Results

6.1 Accuracy Test of Two-Phase Colocalization

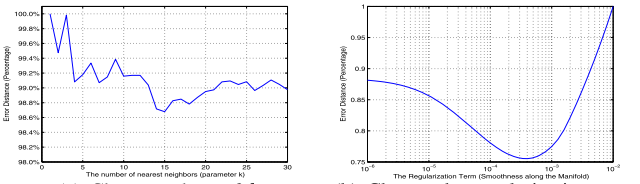
For comparison, we run the following baseline algorithms: 1) LANDMARC, a nearest-neighbor weighting-based method designed for RFID localization [37], 2) Support Vector Regression (SVR), a simplified variant of a kernel-based method used for WSN localization [11], and 3) RADAR, a KNN method for WLAN localization [9]. In each experiment, we randomly picked 500 examples for training and the rest for testing. The training data were further split into labeled and unlabeled parts. The results, shown in Fig. 7, are averaged over 10 repetitions for reducing statistical variability. All results are measured in *relative error distances*, which are error distances in percentage while referring to the maximal error distance in each figure for easy comparison.

LANDMARC, RADAR, and SVR were trained with the labeled part of training data. In contrast, the proposed *two-phase colocalization* method uses both labeled and unlabeled data. We test on two configurations for the *two-phase colocalization* method: 1) "**Colocalization no AP**" uses partially labeled data from mobile devices for training in which we try to recover the locations of the access points,



(a) RFID MD (tags) (b) RFID AP (readers) (c) WSN MD (mobile sensor node)
(d) WSN AP (static sensor nodes) (e) WLAN MD (notebook) (f) WLAN AP (access points)

Fig. 7. Experimental results over 10 repetitions (Mean and Std.): MD for mobile device, AP for access point.



(a) Change values of k (b) Change the regularization term

Fig. 8. Parameter tuning on a validation set.

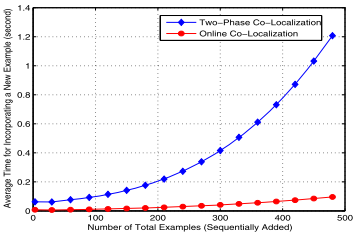


Fig. 9. Average running time comparison.

and 2) "**Colocalization with AP**" repeats the same experiments with the locations of all access points known. Note that all error distances are presented in percentage since they are normalized when referring to the maximal error in each figure.

6.1.1 Model Parameter Setting

Our experiments mainly target showing how labeled and unlabeled data can help increase accuracy and reduce calibration effort in *relative error distance*. We do not specifically fine-tune the parameters. Instead, parameters are determined in a validation set at a coarse level. We set $s^{max} = -30$ dBm and $\sigma = 8$ in the Gaussian function in (1) for signals in all networks. $s^{max} = -30$ is a value that roughly ranks in the top 1 percent of all signal strength values. We avoid using the top first value to avoid potential outliers. We also tried other values in the experiment such as $s^{max} = -40$ and $\sigma = 16$. There was no significant difference in the experiment results. We use k nearest neighbors for building the neighborhood graph in constructing all graph Laplacians. We set $k = 10$ for L_P in (3) and $k = 5$ for L_Q in (4) after trying popular values such as 5, 10, or 15 in manifold learning. γ is a global regularization term for second level terms γ_A, γ_B, γ_C in (8). In the following experiments, we set $\gamma_A = 0.01$, $\gamma_B = 1.0$, $\gamma_C = 0.001$, and $\gamma = 0.0001$, which are tuned in a validation set. The details of a strategy of parameter (γs) tuning and a sensitivity study on the parameters will be described in Section 6.1.3.

6.1.2 Comparison Results

Figs. 7a, 7c, and 7e show the location-estimation errors of different mobile devices by varying the number of labeled examples in a training set whose size is fixed to be 500. We can observe from the figures that first, if we compare the results vertically in each figure, we can see how the unlabeled data help improve the result in the proposed methods. For example, in Fig. 7e, most compared methods have a relative error distance of around 80 percent when using 50 labeled examples. In contrast, the proposed methods have an error of around 40 percent by employing additional 450 unlabeled examples. Second, if we compare the results horizontally in each figure, we can find how our methods reduce calibration effort. For example, in Fig. 7a, most compared methods have a relative error distance of around 60 percent when all 500 examples are labeled. The "**Colocalization with AP**" has similar performance when using only 50 labeled and 450 unlabeled examples. Hence, we save the calibration effort dramatically.

We find that the mobility of the mobile device and the environment complexity were two main factors that affected the performance of the *two-phase colocalization* algorithm. In a static and plane-shaped testbed (Fig. 7a), the radio signals are less noisy and the "**Co-Localization no AP**" configuration demonstrated similar performance as RADAR, LANDMARC, and SVR when the number of labeled examples is small. In a mobile and complex environment, as shown in Fig. 7e, the radio signal is more noisy and the "**Colocalization no AP**" performed much better and more robustly than the compared methods. We have also tried some other combinations of experiments that led to a similar conclusion, such as using RFIDs in a mobile scenario.

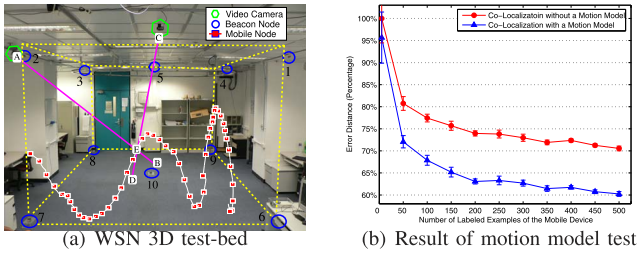
While comparing the results of "**Colocalization no AP**" and "**Colocalization with AP**" in Figs. 7a, 7c, and 7e, we find that knowing the locations of access points is more helpful for localizing the mobile devices in a static and planar scenario (Fig. 7a) than in a mobile and complex environment (see Fig. 7e). Similarly, we can see from Figs. 7b, 7d, and 7f that knowing the locations of mobile devices is more helpful for localizing access points in a static and plane-shaped scenario rather than a mobile and complex environment.

6.1.3 Parameter Sensitivity Test

In this section, we study the parameter sensitivity on the performance of the *two-phase colocalization* algorithm. Fig. 8 shows the sensitivity of relative error distances while varying parameters such as the k for retrieving top nearest neighbors in Fig. 8a and the regularization parameters γ for penalizing the smoothness along the data manifold in Fig. 8b on a validation set. As can be seen from Fig. 8a, the error ranges in $[98.5\%, 99.4\%]$ (less than 1 percent change) when k varies from 5 to 20. This observation is consistent with most manifold-based learning methods when k is generally picked up among popular values such as 5, 10, or 15. Besides the parameter k, there are several γs for controlling the smoothness on data manifolds, γ_A, γ_B, γ_C, and γ. As we described in Section 6.1.1, γ is a global tuning term for γ_A, γ_B, and γ_C in (8). For tuning these parameters in a validation set, we first fix γ_B to 1 and tune the others. The tuning strategy may not be optimal but it works well in our experiments. Fig. 8b shows how the error changes when the global parameter γ ranges in $[10^{-6}\ 10^{-2}]$ while fixing $\gamma_B = 1, \gamma_A = 0.01$, and $\gamma_C = 0.001$. The best value can be picked up in about $[10^{-4}\ 10^{-3}]$. Small changes in parameter setting such as k and γ would not change the trend of the curves shown in Fig. 7.

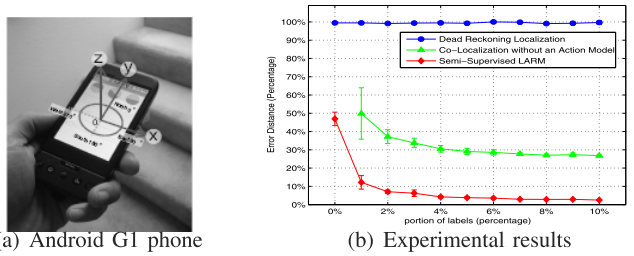
6.2 Speed Test of Online Colocalization

Fig. 9 shows the average running time for adding a new training example. The test was done in Matlab on a computer with a 2.0 GHz CPU. Experimental results show that we can greatly reduce the time for the model adaption



(a) WSN 3D test-bed (b) Result of motion model test

Fig. 10. Experiments with a motion model.



(a) Android G1 phone (b) Experimental results

Fig. 11. Experiments with an action model.

in an online manner. For example, when the training data set size is incrementally enlarged to about 500, the two-phase method needs 1.2 s to reestimate everything, while the online method spends no more than 0.1 s. The *online* method is more than 10 times faster. The localization accuracy of the online model is similar to the two-phase counterpart. The difference is that the neighborhood graph and weight matrix are revised incrementally rather than rebuilt.

6.3 Encode a Motion Model

In this experiment, we aimed to verify that, by employing a Kalman Filter, the error distances of all previous compared methods can be further reduced by 5 to 10 percent in a mobile scenario. Here, we performed an additional three-dimensional tracking experiment, showing the usefulness of encoding motion models. The whole testbed fills up a cubic space of $6.0\ \mathrm{m} \times 6.0\ \mathrm{m} \times 2.0\ \mathrm{m}$. In the testbed, we have 10 static nodes that send out beacon signals. Five of them are deployed on the floor and the rest on the ceiling. There is one more node that moves freely around the environment for tracking experiments. The ground-truth location of the mobile node was exported from four cameras deployed in the laboratory. In Fig. 10a, the location of the mobile node is obtained by computing the intersection point E of lines CD and AB estimated from two different cameras. We collected 1,000 examples, which are split into two parts: 500 examples for training and the rest for testing. Again, we varied the number of labeled examples and repeated the experiments 10 times. The experimental results are shown in Fig. 10b. As can be seen, the error distance of *colocalization* with a motion model is about 10 percent smaller than that without a motion model when sufficient labeled data are available.

6.4 Encode an Action Model

In Section 4.4, we have already demonstrated how LARM works in previous examples shown in Fig. 6. In this section, we conducted experiments to study whether the LARM algorithm can further boost the performance of location tracking by fusing different types of sensors. For this purpose, we used an Android G1 phone for collecting data because it has a lot of built-in sensors such as accelerometer, magnetic sensors, Wi-Fi, etc.

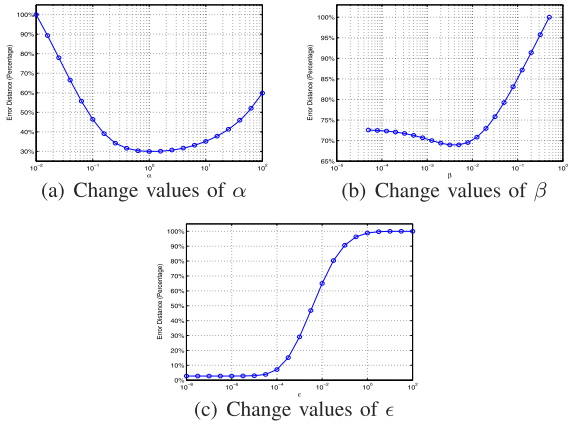
With a built-in three-axis accelerometer sensor, system-detected orientation and direction can be directly read from a built-in compass sensor, and speed values can be indirectly estimated from the accelerometer sensors. Specifically, we applied the estimation method introduced in [34] to infer the speed from accelerometer readings. Since speed and direction values are taken as additional input, in our experiments, we did not explicitly vary the *accuracy* of speed/direction values. Instead, we focused on studying how the LARM algorithm can improve the localization accuracy by embedding an action model even when the speed and direction values contain noise.

The values (unit: $\mathrm{m/s^2}$) are in the format of (X, Y, Z). The X-axis refers to the screen's horizontal axis (the small edge in portrait mode, the long edge in landscape mode) and points to the right. The Y-axis refers to the screen's vertical axis and points toward the top of the screen. The Z-axis points toward the sky when the device is lying on its back on a table. Fig. 11a shows the directions of X, Y, and Z when the phone works in portrait mode. An accelerometer sensor can be used as a pedometer for speed estimation. Orientation is estimated from a built-in compass sensor. It is represented by a triple $(Azimuth, Pitch, Roll)$. All values are angles in degrees. $Azimuth$ is the rotation around the Z-axis ($0° \leq azimuth < 360°$), for which $0° = \mathrm{North}$, $90° = \mathrm{East}$, $180° = \mathrm{South}$, $270° = \mathrm{West}$, as shown in Fig. 11a. $Pitch$ is the rotation around the X-axis ($-180 \leq pitch \leq 180$), with positive values when the Z-axis moves toward the Y-axis. $Roll$ is the rotation around the Y-axis ($-90 \leq roll \leq 90$), with positive values when the Z-axis moves toward the X-axis.

Alignment among sensors is necessary during data collection because they have different sample rates. The rate of sampling Wi-Fi signal strength is 2 Hz. Accelerometer and orientation sensors have a sample rate of 45 Hz and thus will be downsampled after action recognition naturally. In the area shown in Fig. 6a, we walked around and collect 1,000 examples at sample rate 2 Hz.

6.4.1 Overall Results

In each experiment, we randomly picked a small portion of data for labeling with the rest for testing. The results (mean error and standard deviation) shown in Fig. 11b are averaged over 10 repetitions. The horizontal axis is the percentage of data that are labeled, which ranges from 0 to 10 percent. The vertical axis is the average error distance in meters. As can be seen, Dead Reckoning Localization without using any labeled data has a large error due to drifting factor. When we combine Wi-Fi tracking through colocalization and Dead Reckoning Localization together using *unsupervised* LARM, the error is reduced at least by half. If some small percent of labeled data are available, the location-estimation error of the *semi-supervised* LARM algorithm can be reduced significantly. The fusion of Wi-Fi and motion sensors with LARM also has better performance than using partially labeled Wi-Fi data alone (denoted "Colocalization without an Action Model").



(a) Change values of α (b) Change values of β

(c) Change values of ϵ

Fig. 12. Parameter sensitivity study with an action model.

6.4.2 Parameter Sensitivity Test

Compared to the *two-phase colocalization* algorithm, the action-model-embedded algorithm LARM has three more parameters, α, β, and ϵ. In experiments shown in the previous section, we use the same parameter settings for γ_A, γ_B, γ_C, γ, and k as described in Section 6.1.3, and set $\alpha = 0.5$, $\beta = 0.001$, and $\epsilon = 10^{-6}$ in (16), which are all tuned in a validation set. In this section, we fix the values of γ_A, γ_B, γ_C, γ, and k to test the sensitivity at different values of α, β, and ϵ on the overall performance of LARM. Fig. 12 shows how the error distance changes while looping through α, β, and ϵ. As can be seen in Fig. 12a, LARM performs well when α ranges in $[10^{-1}\ 10^{1}]$. Similarly, Fig. 12b suggests that the best value for β falls in the range $[10^{-4}\ 10^{-2}]$. Fig. 12c shows that the performance of LARM is good and varies little when ϵ is set to a value smaller than 10^{-4}.

7 Conclusion and Future Works

In this paper, we proposed a novel semi-supervised graph Laplacian approach to solve the problem of simultaneously recovering the locations of both mobile devices and access points. In our colocalization framework, we estimated the relative locations of mobile devices and access points by exploiting an SVD-based method, and estimated the absolute locations using a small collection of labeled data through graph Laplacian methods. Our extensive experiments in three different testbeds showed that we can achieve high performance with much less calibration effort as compared to several previous approaches. Meanwhile, our model can deal with data stream and adjust itself *online* relatively faster while compared to its two-phase counterpart. Finally, we extend our framework for multiple sensor fusion via an Action Respecting Manifold. Several demonstrations and experimental results show that the performance of combing multiple sensors for localization is much better than using them individually.

The significance of the work is that we can leverage the knowledge of the access point locations, the mobile device trajectories, and motion sensors to obtain more accurate localization. We will continue to evaluate the performance in a large-scale and dynamic environment, e.g., in a city level and at different time periods. We may also vary more parameters such as the number of access points and their deployment density and study the robustness of our proposed algorithm.

Acknowledgments

The authors thank Seth Teller, Rong Pan, and Vincent Wenchen Zheng for their helpful discussions and ideas during the long run of this work in the past years. They also thank the support of Hong Kong RGC/NSFC N_HKUST624/09 and Hong Kong RGC grant 621010, and Microsoft Research Asia Grant MRA10EG01.

References

[1] L. Liao, D.J. Patterson, D. Fox, and H.A. Kautz, "Learning and Inferring Transportation Routines," *Artificial Intelligence*, vol. 171, nos. 5/6, pp. 311-331, 2007.

[2] Y. Zheng and X. Xie, "Learning Travel Recommendations from User-Generated GPS Traces," *ACM Trans. Intelligent Systems and Technology*, vol. 2, pp. 2:1-2:29, Jan. 2011.

[3] K. Farrahi and D. Gatica-Perez, "Discovering Routines from Large-Scale Human Locations Using Probabilistic Topic Models," *ACM Trans. Intelligent Systems and Technology*, vol. 2, pp. 3:1-3:27, Jan. 2011.

[4] M.A. Batalin, G.S. Sukhatme, and M. Hattig, "Mobile Robot Navigation Using a Sensor Network," *Proc. IEEE Int'l Conf. Robotics and Automation*, pp. 636-642, Apr. 2004.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, Sept. 2005.

[6] A. Kotanen, M. Hannikainen, H. Leppakoski, and T.D. Hamalainen, "Positioning with IEEE 802.11b Wireless LAN," *Proc. 14th IEEE Indoor and Mobile Radio Comm.*, pp. 2218-2222, Sept. 2003.

[7] Q. Yang, S.J. Pan, and V.W. Zheng, "Estimating Location Using Wi-Fi," *IEEE Intelligent Systems*, vol. 23, no. 1, pp. 8-13, Jan./Feb. 2008.

[8] D. Maligan, E. Elnahrawy, R. Martin, W. Ju, P. Krishnan, and A.S. Krishnakumar, "Bayesian Indoor Positioning Systems," *Proc. IEEE INFOCOM*, pp. 1217-1227, Mar. 2005.

[9] P. Bahl and V. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," *Proc. IEEE INFOCOM*, vol. 2, pp. 775-784, Mar. 2000.

[10] B. Ferris, D. Hahnel, and D. Fox, "Gaussian Processes for Signal Strength-Based Location Estimation," *Proc. Robotics: Science and Systems*, Aug. 2006.

[11] X. Nguyen, M.I. Jordan, and B. Sinopoli, "A Kernel-Based Learning Approach to Ad Hoc Sensor Network Localization," *ACM Trans. Sensor Networks*, vol. 1, no. 1, pp. 134-152, 2005.

[12] S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman, "Indexing by Latent Semantic Analysis," *J. Am. Soc. Information Science*, vol. 41, no. 6, pp. 391-407, 1990.

[13] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Computation*, vol. 15, no. 6, pp. 1373-1396, 2003.

[14] J. Ham, D. Lee, and L. Saul, "Semisupervised Alignment of Manifolds," *Proc. 10th Int'l Workshop Artificial Intelligence and Statistics*, pp. 120-127, Jan. 2005.

[15] J.J. Pan and Q. Yang, "Co-Localization from Labeled and Unlabeled Data Using Graph Laplacian," *Proc. 20th Int'l Joint Conf. Artificial Intelligence*, pp. 2166-2171, 2007.

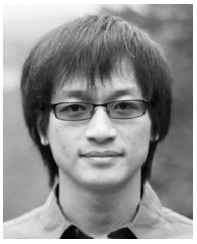
[16] M.H.C. Law and A.K. Jain, "Incremental Nonlinear Dimensionality Reduction by Manifold Learning," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, pp. 377-391, Mar. 2006.

[17] O. Kouropteva, O. Okun, and M. Pietikainen, "Incremental Locally Linear Embedding Algorithm," *Pattern Recognition*, vol. 38, no. 10, pp. 1764-1767, Oct. 2005.

[18] J.J. Pan, Q. Yang, and S.J. Pan, "Online Co-Localization in Indoor Wireless Networks by Dimension Reduction," *Proc. 22nd Nat'l Conf. Artificial Intelligence*, pp. 1102-1107, 2007.

[19] B. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM Using Gaussian Process Latent Variable Models," *Proc. 20th Int'l Joint Conf. Artificial Intelligence*, pp. 2480-2485, 2007.

[20] T. Yairi, "Map Building without Localization by Dimensionality Reduction Techniques," *Proc. 24th Int'l Conf. Machine Learning*, pp. 1071-1078, 2007.
[21] E. Foxlin, "Pedestrian Tracking with Shoe-Mounted Inertial Sensors," *IEEE Computer Graphics and Applications*, vol. 25, no. 6, pp. 38-46, Nov./Dec. 2005.
[22] M. Bowling, A. Ghodsi, and D. Wilkinson, "Action Respecting Embedding," *Proc. 22nd Int'l Conf. Machine Learning*, pp. 65-72, 2005.
[23] J. Letchner, D. Fox, and A. LaMarca, "Large-Scale Localization from Wireless Signal Strength," *Proc. 20th Nat'l Conf. Artificial Intelligence*, pp. 15-20, July 2005.
[24] M. Youssef, A. Agrawala, and U. Shankar, "WLAN Location Determination via Clustering and Probability Distributions," *Proc. First IEEE Int'l Conf. Pervasive Computing and Comm.*, pp. 143-150, Mar. 2003.
[25] S.J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. Knowledge and Data Eng.*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.
[26] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge Univ. Press, 2004.
[27] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples," *J. Machine Learning Research*, vol. 7, pp. 2399-2434, Nov. 2006.
[28] J.J. Pan, Q. Yang, H. Chang, and D.Y. Yeung, "A Manifold Regularization Approach to Calibration Reduction for Sensor-Network Based Tracking," *Proc. 21st Nat'l Conf. Artificial Intelligence*, pp. 988-993, July 2006.
[29] F. Chung, *Spectral Graph Theory*. Am. Math. Soc., 1997.
[30] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888-905, Aug. 2000.
[31] I.S. Dhillon, "Co-Clustering Documents and Words Using Bipartite Spectral Graph Partitioning," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 269-274, 2001.
[32] B. Hendrickson, "Latent Semantic Analysis and Fiedler Embeddings," *Proc. SIAM Workshop Text Mining*, Apr. 2006.
[33] X. Zhu, Z. Ghahramani, and J.D. Lafferty, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions," *Proc. 20th Int'l Conf. Machine Learning*, pp. 912-919, Aug. 2003.
[34] N. Ravi, N. Dandekar, P. Mysore, and M.L. Littman, "Activity Recognition from Accelerometer Data," *Proc. Seventh Innovative Applications of Artificial Intelligence Conf.*, pp. 1541-1546, 2005.
[35] E.O. Brigham and R.E. Morrow, "The Fast Fourier Transform," *IEEE Spectrum*, vol. 4, no. 12, pp. 63-70, Dec. 1967.
[36] J.A. Ward, P. Lukowicz, and H.W. Gellersen, "Performance Metrics for Activity Recognition," *ACM Trans. Intelligent Systems and Technology*, vol. 2, pp. 6:1-6:23, Jan. 2011.
[37] L. Ni, Y. Liu, Y. Lau, and A. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," *Proc. First IEEE Int'l Conf. Pervasive Computing and Comm.*, pp. 407-416, Mar. 2003.



Jeffrey Junfeng Pan received the BS degree in computer science from Sun Yat-sen University in China in 2003, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2008. He is currently a research scientist at Facebook Inc., Palo Alto, California. He was a research engineer at Google Inc., Mountain View, California, from 2007 to 2010. His main research interests include machine learning, data mining, and their applications in ads optimization and mobile computing. He is a member of the IEEE.

Sinno Jialin Pan received the BS and MS degrees in applied mathematics from Sun Yat-sen University in China in 2003 and 2005, respectively, and the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology in 2010. He is currently a research fellow at the Institute for Infocomm Research, Singapore. His main research interests include transfer learning, semi-supervised learning, and their applications in pervasive computing and information extraction. He is a member of the IEEE.

Jie Yin received the BE degree from Xi'an Jiaotong University, China, in 2001, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2006. She is currently a research scientist in the Information Engineering Laboratory at the CSIRO ICT Centre, Australia. Her research interests include data mining, machine learning, and their applications to sensor-based activity recognition, social network analysis, and web mining. She is a member of the IEEE.

Lionel M. Ni received the PhD degree in electrical and computer engineering from Purdue University, West Lafayette, Indiana, in 1980. He is the chair professor in the Computer Science and Engineering Department at the Hong Kong University of Science and Technology (HKUST). He served as the department head from 2002 to 2008. He also serves as the special assistant to the President at HKUST, the dean of the HKUST Fok Ying Tung Graduate School, and the director of the HKUST China Ministry of Education/Microsoft Research Asia IT Key Lab. He was the chief scientist of the National Basic Research Program of China (973 Program) on Wireless Sensor Networks. He has chaired many professional conferences and has received a number of awards for authoring outstanding papers. He is a fellow of the IEEE.

Qiang Yang received the BS degree from Peking University in astrophysics and the PhD degree in computer science from the University of Maryland, College Park. He is a professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests are data mining and artificial intelligence. He was elected a vice chair of ACM SIGART in July 2010. He is the founding editor in chief of the *ACM Transactions on Intelligent Systems and Technology* (*ACM TIST*). He was a PC cochair or general cochair for a number of international conferences, including ACM KDD '10, ACM IUI '10, etc. He is a fellow of the IEEE.