Discriminant Minimization Search for Large-Scale RF-Based Localization Systems

Sheng-Po Kuo and Yu-Chee Tseng

Abstract—In large-scale fingerprinting localization systems, fine-grained location estimation and quick location determination are conflicting concerns. To achieve finer grained localization, we have to collect signal patterns at a larger number of training locations. However, this will incur higher computation cost during the pattern-matching process. In this paper, we propose a novel *discriminant minimization search (DMS)*-based localization methodology. Continuous and differentiable discriminant functions are designed to extract the *spatial correlation* of signal patterns at training locations. The advantages of the DMS-based methodology are threefold. First, with through slope of discriminant functions, the exhaustive pattern-matching process can be replaced by an optimization search process, which could be done by a few quick jumps. Second, the continuity of the discriminant functions helps predict signal patterns at untrained locations so as to achieve finer grained localization. Third, the large amount of training data can be compressed into some functions that can be represented by a few parameters. Therefore, the storage space required for localization can be significantly reduced. To realize this methodology, two algorithms, namely, *Newton-PL* and *Newton-INT*, are designed based on the concept of *gradient descent search*. Simulation and experiment studies show that our algorithms do provide finer grained localization and incur less computation cost.

Index Terms—Discriminant function, fingerprinting localization, gradient descent search, mobile computing, pattern-matching localization, wireless network.

1 Introduction

Location tracking is a critical issue in *location-based services (LBSs)*. Although GPS [1] has been widely used, it has some inherent limitations, such as unavailability in indoor environments and accuracy concerns. Therefore, many techniques have been developed by relying on existing wireless or telecommunication infrastructures to compensate the drawbacks of GPS [2], [3]. The *cell-based* approach [4] is easy to implement. It is suitable for large-scale environments and does not incur extra infrastructure cost. However, such systems typically have accuracy concerns. The *multilateration* approach [4] is based on some parametric (propagational) models to measure distances from signal sources. Such systems are more accurate when the fading effect can be modeled accurately. However, in most complex environments, such as indoor environments, the fading effect is hard to predict.

In this work, we are interested in *pattern-matching* localization based on the fingerprinting approach, such as RADAR [5]. Unlike those localization methods based on parametric models, such systems do not rely on preknowledge of the locations of signal sources and do not rely on calculating the signal fading factor. Instead, such systems rely on a *training phase* to learn the received signal strength (RSS) patterns at a set of training locations from those signal sources [5], [6], [7], [8], [9], [10], [11], [12]. These signal sources, or base stations, can be existing infrastructures, such as IEEE 802.11, GSM, or WiMAX networks. In the *positioning phase*, to locate a mobile device, the RSS pattern measured by the mobile device is compared against the data collected in the training phase. Such pattern-matching process usually incurs a lot of computational cost. Concerns of the pattern-matching schemes include the grain size of training locations and the time complexity in the positioning phase. Recently, some works have addressed the grain size issue [10], [13], [14], [15], [16], [17], [18], [19], [20]. Naively increasing the number of training locations is not scalable because both the training and positioning overheads will increase significantly. To reduce the grain size of training locations, some numerical methods are proposed to infer some *virtual* training data from real training data. Hence, the training overhead can be limited to a relatively small number of real training locations. However, the positioning overhead is still proportional to the total number of real and virtual training locations. The authors in [11], [21], [22], [23] try to reduce the positioning cost by partitioning the training data into several subsets, each characterized by a *metafeature vector*. Therefore, most subsets can be eliminated after examining their metafeature vectors. However, the grain size issue is not addressed.

Finer grained location estimation and quicker location determination are typically conflicting concerns in fingerprinting localization systems. As far as we know, there is no existing solution that can meet both goals at the same time. In this paper, we propose a novel *discriminant minimization search (DMS)*-based localization methodology, which relies on a continuous and differentiable *discriminant function* obtained by some numerical methods to stand for those

- *S.-P. Kuo is with Telcordia Applied Research Center, Suite E-453, 4F, No. 19-13, SanChung Rd., Taipei 115, Taiwan. E-mail: kuopo@research.telcordia.com.*
- *Y.-C. Tseng is with the Department of Computer Science, National Chiao Tung University, Hsin-Chu 30010, Taiwan, and the Department of Information and Computer Engineering, Chung-Yuan Christian University, Taiwan. E-mail: yctseng@cs.nctu.edu.tw.*

Manuscript received 24 Apr. 2009; revised 6 Dec. 2009; accepted 17 Mar. 2010; published online 26 Aug. 2010.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2009-04-0140.
Digital Object Identifier no. 10.1109/TMC.2010.67.

discrete training data. In our design, the global minimizer of the discriminant function is expected to be the real location of the mobile device. Since the function is continuous, finer location estimation can be achieved in an easier manner. Also, the differentiability property of the discriminant function provides slope information, and thus, a search direction in the search space. This can greatly speed up the positioning process for finding an optimizer by a few quick jumps.

Relying on the concept of *gradient descent search*, we design two localization algorithms, called *Newton-PL* and *Newton-INT*. They differ in their definitions of discriminant functions. The former is based on a path loss model and the latter is based on an interpolation technique. The Newton-PL algorithm is close to a parametric positioning method and has two main features. First, it is much quicker. Second, it can extract a few scalars from the training data to represent the gradients that are needed during the search process. In fact, these scalars can be computed offline. So, the large amount of the training data can be significantly reduced to a few scalars. These features make Newton-PL very suitable for implementation on those resource-limited portable devices. On the other hand, the Newton-INT algorithm is close to a pattern-matching technique and is more complicated in its search process than Newton-PL. However, it is still more efficient as compared to the classic pattern-matching techniques and is more resilient to the unpredictable signal fading problem in indoor environments than the parametric positioning techniques. Hence, it is quite suitable for indoor applications.

To evaluate the performance of the proposed algorithms, we have conducted both simulations and experiments. In our simulations, we adopt the *radio irregularity model (RIM)* [24], which has been shown to be able to better reflect the physical reality, such as the influence of hardware difference, nonisotropic propagation, and dynamic signal fading effect. Besides, we further modify RIM to simulate signal attenuation caused by obstacles. These features, which make the indoor positioning problem more difficult, are used to test our schemes. In our simulation study, we tune the parameters of RIM to evaluate Newton-PL and Newton-INT under different environmental conditions and investigate the issues of training grain size, initial point selection, and the cost for building positioning models. On the other hand, we also implement a median-scale real environment to verify the correctness of the proposed algorithms by varying factors such as training location density and user moving speed. These results prove that our proposed algorithms do achieve finer grained and efficient localization simultaneously.

The rest of this paper is organized as follows: Section 2 gives some preliminaries of fingerprinting localization. Section 3 briefly presents the prior arts for finer grained or efficient localization. The proposed DMS-based localization algorithms are in Section 4. Performance studies are in Section 5. Finally, Section 6 draws our conclusions.

2 Preliminaries

A fingerprinting localization system generally works as follows: We are given a set of *base stations* $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$, each capable of transmitting radio signals periodically in a field $\mathcal{F} \subseteq \mathbb{R}^2$, and a set of known training locations $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$ also in $\mathcal{F}$. Each location ℓ_i is labeled by a 2D Cartesian coordinate (x_i, y_i). The system works in two phases. In the *training phase*, at each training location ℓ_i, $i = 1..m$, we measure the signal strengths from base stations for a period of time and create a *feature vector* $\upsilon_i = [\upsilon_{i,1}, \upsilon_{i,2}, \ldots, \upsilon_{i,n}]$ for ℓ_i, where $\upsilon_{i,j} \in \mathbb{R}$ is the averaged RSS from b_j, $j = 1..n$. The feature space of υ_i is written as $\mathcal{S} \subseteq \mathbb{R}^n$. The set of feature vectors is collected in a database $\mathcal{V} = \{\upsilon_1, \upsilon_2, \ldots, \upsilon_m\}$. In the *positioning phase*, a mobile device can estimate its location in $\mathcal{F}$ by measuring its RSS vector $\boldsymbol{s} = [s_1, s_2, \ldots, s_n]$ and comparing $\boldsymbol{s}$ against $\mathcal{V}$. The positioning process can be modeled by a mapping function $\mathrm{loc} : (\mathcal{L}, \mathcal{V}, \boldsymbol{s}) \mapsto \mathcal{F}$, whose goal is to determine the location of the mobile device in $\mathcal{F}$. If the returned location is within a finite number of locations (such as those in $\mathcal{L}$), we regard loc as a *discrete* positioning function. For example, in [5], given $\boldsymbol{s}$, the *Nearest Neighbor in Signal Space (NNSS)* algorithm suggests a distance function h:

$$h(\ell_i) = \|\boldsymbol{s}, \upsilon_i\| = \sum_{j=1}^{n} \sqrt{(s_j - \upsilon_{i,j})^2}. \tag{1}$$

Then, the positioning result $\mathrm{loc}(\mathcal{L}, \mathcal{V}, \boldsymbol{s}) = \arg\min_{\ell_i \in \mathcal{L}} h(\ell_i)$. In NNSS, the result is obtained through an exhaustive search, i.e., every $\upsilon_i \in \mathcal{V}$ is examined. On the contrary, if the returned location can be any one in $\mathcal{F}$, we regard loc as a *continuous* positioning function. The proposed DMS-based localization belongs to the continuous category.

3 Related Works

Most fingerprinting localization systems adopt a discrete positioning function. The accuracy of estimated locations thus depends on the density of training locations. To achieve finer grained localization after training data have been collected, one may try to generate some virtual training locations from real ones. There are two approaches. The first one is to build propagational models [5], [13], [14], [15]. In [5], [15], an open space propagational model considering wall attenuation factor is proposed to estimate the signal patterns at each virtual training location. However, these models do not consider the effects of signal reflection. Hence, to more accurately estimate signal attenuation caused by distinct features in an indoor environment, a hidden environment model is proposed in [14], which can measure an adjustment for each virtual training location according to the spatial relationship between itself and the surrounding training data. In [13], using building layout and structure, a ray-tracing approach is proposed to derive signal propagation considering various possible transmission paths, including direct and indirect ones. However, such approaches are still too simple to predict detailed signal fading and propagation in small areas for positioning purpose.

The second approach is by interpolation, such as linear interpolation [16], [19], [25], Akima splines interpolation [17], [20], and Shepard interpolation [18]. Through interpolation, we can more precisely guess the signal patterns at those untrained locations by referring to nearby training locations. However, prior arts related to interpolation do

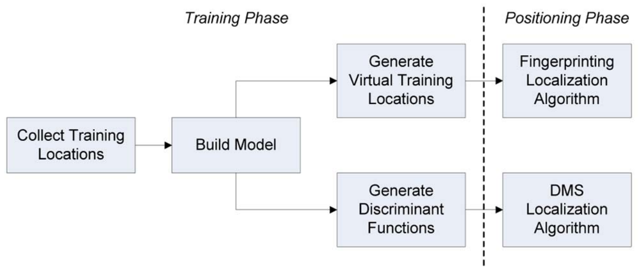
Training Phase
Positioning Phase
Collect Training Locations
Build Model
Generate Virtual Training Locations
Generate Discriminant Functions
Fingerprinting Localization Algorithm
DMS Localization Algorithm

Fig. 1. Comparison of interpolation strategies.

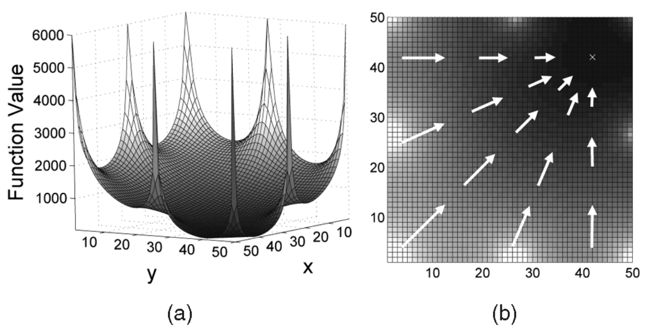

Fig. 2. (a) The surface view and (b) the top view of the function $h(\ell_i)$ for all $\ell_i \in \mathcal{L}$ given s measured at (42,42). White arrows denote potential search directions. Longer arrows mean quicker jumps, while shorter arrows mean finer searches.

not consider the computation and storage requirements incurred at the runtime. In this work, we do not try to generate individual virtual training data. Instead, we define discriminant functions to represent the search space. Fig. 1 compares our approach against existing ones. Our DMS-based localization algorithm uses the model directly, so the required amount of storage space can be much reduced. As to be shown later on, it also significantly alleviates the computation cost in the positioning phase. Therefore, our scheme is more suitable for large-scale LBSs.

On the other hand, to reduce the pattern-matching cost, several solutions have been proposed to partition a large training set into smaller *clusters* [11], [21], [22], [26]. Hence, the computational cost can be restricted to searching a smaller set. In [22], training locations that see the same set of base stations with the strongest signal strengths are grouped together. This technique is easy to be implemented, but it is hard to predict how many clusters will be generated. In [11], [21], [23], clustering is done by the k-means algorithm, a widely used technique to partition a set of objects through an iterative process. The search effort is then reduced to identifying a proper cluster and examining only the members of that cluster (such as by a decision tree [21]). In [26], training locations are clustered according to their physical locations. Hence, in a continuous tracking scenario, location search can be restricted in a set of clusters, including the current cluster and its neighboring ones. In addition, the clustering technique is also used in improving positioning accuracy [23], which proposes to group those more likely training locations together and choose their centroid as the estimated location. Observing that building clusters also takes time, Kushki et al. [27] propose not to cluster in the training phase. Instead, it proposes a spatial filtering technique to eliminate those training locations too far away from the current RSS pattern s in the positioning phase. The filtering operation does a rough evaluation between s and each v_i. This rough evaluation could simply be the number of different base stations and is thus quite time-saving. After pruning off most inadequate training locations, we can conduct a more detailed pattern-matching procedure. Compared to the clustering approach, this spatial filtering is still more expensive computationally, but can better adapt to dynamic environments. Our DMS methodology has the advantages of both approaches. Through simulation studies, we will show that our approach incurs even lower computation cost than clustering in large-scale environments. Further, our model can easily be rebuilt or modified as the environments change.

4 Discriminant Minimization Search-Based Localization

To motivate our work, we observe that many existing pattern-matching localization algorithms [5], [6], [7], [12], [15], [17], [18], [25], [28], [29] try to determine locations by exhaustively searching the set $\mathcal{V}$. This could be very time-consuming when $\mathcal{V}$ is large. For example, in a wireless city, thousands or millions of training locations may have to be searched. To relieve this problem, we may exploit the *spatial correlation* of the data set in $\mathcal{V}$ in the sense that training locations in proximity should have similar feature vectors, while those that are farther away should have a higher degree of difference. For example, Fig. 2 draws $h(\ell_i)$ for 2,500 training locations in a 50×50 grid field with eight base stations around the boundary and s is the RSS at location $(42, 42)$. We can see that from any location in the field, $h(\ell_i)$ is smoothly decreasing as ℓ_i is closer to $(42, 42)$. Intuitively, if we use $h(\ell_i)$ as the degree of difference between s and v_i, then the slope between two ℓ_i and ℓ_j can guide our search direction. By exploiting gradient search, we can proceed with some quicker jumps in areas with steeper slopes, and when it is closer to the target location with gentler slopes, finer searches can be conducted. Based on this concept, we design our scalable DMS-based localization methodology.

The basic idea behind our DMS-based localization is to construct a *continuous* and *differentiable* discriminant function $f : \mathcal{F} \mapsto \mathbb{R}^+$ from the given $\mathcal{L}$, $\mathcal{V}$, and s. We then apply an optimization search algorithm to find the location in $\mathcal{F}$ that minimizes f. The search algorithm is expected to quickly find a location $\ell_{est} \in \mathcal{F}$ that is close to the optimal location $\ell^* = \arg\min_{\ell \in \mathcal{F}} f(\ell)$. Specifically, we will exploit the *gradient descent search* [30] to find the optimizer of the discriminant function. This scheme is an iterative process which can quickly converge to the minimizer of the given discriminant function. We propose two possible ways to define the discriminant function: one based on a path loss model and the other based on an interpolation technique. However, both have the continuity and the differentiability properties.

4.1 Gradient Descent Search

The gradient descent search is a well-known technique to find the optimizer of a given continuous and differentiable objective function [30]. This technique is an iterative search process. In each iteration, we try to get closer to the optimizer. Initially, a starting location $\ell^{(0)}$ is randomly selected. Then, we compute $\ell^{(i+1)}$, $i \geq 0$,

$$\ell^{(i+1)} = \ell^{(i)} + \alpha d^{(i)}, \tag{2}$$

where α is a constant scalar and $d^{(i)}$ is a vector in $\mathcal{F}$. Generally, the gradient descent search is terminated when one of the following two conditions is satisfied: 1) the absolute difference of two successive results is less than a predefined threshold $\Delta\ell_{min}$, that is, $\|\ell^{(i+1)} - \ell^{(i)}\| < \Delta\ell_{min}$, and 2) the number of iterations reaches a predefined bound i_{max}. The terminated point, written as ℓ_{est}, is regarded as the minimizer ℓ^* of f.

There are several practices to determine a suitable direction $d^{(i)}$ for each iteration i. One efficient search technique is *Newton's method* [30]. It iteratively uses the minimizers of quadratic functions to approach the minimizer of the original objective function. It assumes that the first and second derivatives of the objective function can be obtained. Hence, we can calculate $f(\ell^{(i)})$, its gradient

$$\nabla f(\ell^{(i)}) = \left[\frac{\partial}{\partial x} f(\ell^{(i)}),\ \frac{\partial}{\partial y} f(\ell^{(i)})\right]^T, \tag{3}$$

and its Hessian matrix

$$\nabla^2 f(\ell^{(i)}) = \begin{bmatrix} \frac{\partial^2}{\partial x^2} f(\ell^{(i)}) & \frac{\partial^2}{\partial x \partial y} f(\ell^{(i)}) \\ \frac{\partial^2}{\partial y \partial x} f(\ell^{(i)}) & \frac{\partial^2}{\partial y^2} f(\ell^{(i)}) \end{bmatrix}, \tag{4}$$

for any given coordinate $\ell^{(i)} = (x^{(i)}, y^{(i)})$. According to the discriminant function f, given $\ell^{(i)}$ in the ith iteration, we find a quadratic function $q_i(\ell)$ to approximate $f(\ell)$ such that $f(\ell^{(i)}) = q_i(\ell^{(i)})$, $\nabla f(\ell^{(i)}) = \nabla q_i(\ell^{(i)})$, and $\nabla^2 f(\ell^{(i)}) = \nabla^2 q_i(\ell^{(i)})$.

With these constraints, instead of minimizing f directly, we can minimize functions q_i because they have the same first-order necessary condition and second-order sufficient condition [30]. In general, we can define

$$\begin{aligned} q_i(\ell) = f(\ell^{(i)}) &+ \nabla f(\ell^{(i)})(\ell - \ell^{(i)}) \\ &+ \frac{1}{2}\nabla^2 f(\ell^{(i)})(\ell - \ell^{(i)})^2. \end{aligned} \tag{5}$$

According to the first-order necessary condition, q_i's minimizer appears when

$$\nabla q_i(\ell) = \nabla f(\ell^{(i)}) + \nabla^2 f(\ell^{(i)})(\ell - \ell^{(i)}) = 0. \tag{6}$$

To satisfy this condition, we let $\ell = \ell^{(i+1)}$. Hence,

$$\begin{aligned} \ell^{(i+1)} &= \ell^{(i)} - \frac{\nabla f(\ell^{(i)})}{\nabla^2 f(\ell^{(i)})} \\ &= \ell^{(i)} - \nabla^2 f(\ell^{(i)})^{-1} \nabla f(\ell^{(i)}). \end{aligned} \tag{7}$$

The value of $\ell^{(i+1)}$ is the approximation function q_i's minimizer. However, remember that our goal is to estimate the minimizer of f. So, we need to verify if this value is close enough to f's actual minimizer ℓ^* or not. If the termination conditions are not satisfied, we generate another $q_{i+1}(\ell)$ at $\ell^{(i+1)}$ and repeat this process again. Note that the Hessian matrix $\nabla^2 f(\ell^{(i)})$ may not be positive definite, so the search direction may not point to a descent direction, implying that the search process may be divergent. Here, we use the Levenberg-Marquardt modification [31] to avoid this situation. The idea is quite intuitive: if the Hessian matrix is not positive definite, we add a small quantity $\mu_i I$ to it, where I is an identity matrix and μ_i is a sufficiently large scalar to make $\nabla^2 f(\ell^{(i)}) + \mu_i I$ positive definite. To sum up, comparing (7) against (2), we define the search direction as

$$d^{(i)} = -(\nabla^2 f(\ell^{(i)}) + \mu_i I)^{-1} \nabla f(\ell^{(i)}). \tag{8}$$

Generally, the step size α can be changed in every iteration. However, deciding a good step size is time-consuming.[1] Hence, we simply use a fixed $\alpha = 1$ here.

In our localization problem, the discriminant function $f(\ell)$ will represent the degree of dissimilarity between s and the feature vector of location $\ell \in \mathcal{F}$. Hence, the minimizer of f will be the estimated location. To apply the gradient descent search algorithm, this function is required to be continuous and differentiable. The gradient can be interpreted as the spatial correlation of locations. Via the direction of gradient, we can quickly estimate the most likely location ℓ^* of the observed signal s. Here, it is worth noting the *local minimum problem* during our search for the global minimum. Fortunately, this can be effectively avoided by selecting a good initial point. We will discuss this issue in Section 4.4. Bellow, we introduce two designs of f.

4.2 Newton's Method Using Path Loss Model (Newton-PL)

This design tries to estimate the channel fading. It intends to model the amount of signal degradation, i.e., path loss, from each base station at each location in $\mathcal{F}$. In the training phase, some path-loss-related parameters will be computed. In the positioning phase, a discriminant function will be constructed according to these parameters and the measured s. From this function, we estimate ℓ_{est} as stated in Section 4.1.

We adopt the *log-distance path loss model* [32] to formulate signal propagation. The path loss at a distance of d can be written as

$$PL(d) = PL(d_0) + 10\phi \log\left(\frac{d}{d_0}\right), \tag{9}$$

where d_0 is the reference distance (here, we set $d_0 = 1$) and ϕ is the pass loss exponent. Hence, the RSS of b_j at ℓ can be expressed by

$$\begin{aligned} P_r(\ell, b_j) &= P_t - PL(\|\ell, b_j\|) \\ &= P_{ref} - 10\phi \log(\|\ell, b_j\|), \end{aligned} \tag{10}$$

where P_t is the transmission power, $P_{ref} = P_t - PL(d_0)$ is the reference power, and $\|\ell, b_j\|$ denotes the euclidean

1. Some sophisticated strategies can guarantee the *descent property* by an additional 1D optimization search to obtain a better step size [30] (i.e., it satisfies $f(\ell^{(i+1)}) < f(\ell^{(i)})$ in each iteration i).

distance between ℓ and b_j in $\mathcal{F}$. To obtain $\|\ell, b_j\|$, the location of b_j should be known. This could be obtained by configuration or by inference from training data [33].

In (10), P_{ref} and ϕ are unknown environment and hardware-dependent factors. We propose to estimate their values for each base station b_i independently. Specifically, let P^j_{ref} and ϕ_j be these factors for b_j. For each training location ℓ_i, $i = 1..m$, it is expected that

$$P^j_{ref} - 10\phi_j \log(\|\ell_i, b_j\|) = \upsilon_{i,j}. \tag{11}$$

Let $\mathbf{x} = [P^j_{ref}, \phi_j]^T$. Putting (11) for $i = 1..m$ together,

$$\underbrace{\begin{bmatrix} 1 & -10\log(\|\ell_1, b_j\|) \\ \vdots & \vdots \\ 1 & -10\log(\|\ell_m, b_j\|) \end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix} P^j_{ref} \\ \phi_j \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \upsilon_{1,j} \\ \vdots \\ \upsilon_{m,j} \end{bmatrix}}_{C}. \tag{12}$$

By the least-squares analysis [30], $\mathbf{x}$ can be measured as

$$\mathbf{x} = \left[P^j_{ref}, \phi_j\right]^T = (A^TA)^{-1}A^TC. \tag{13}$$

Note that when $\|\ell_i, b_j\|$ is too large, it makes no or little sense to add the corresponding equation into (12). In this case, the corresponding ith row in A and C can be removed from (12). Also, note that the above computation is independent of s and can be done at the training stage.

Given s, we define the discriminant function as

$$f(\ell) = \sum_{j=1}^{n} (s_j - P_r(\ell, b_j))^2. \tag{14}$$

Note that in (14), ℓ is a variable in $\mathcal{F}$. To compute (2), we derive

$$\frac{\partial}{\partial x} f(\ell) = (-2)\sum_{j=1}^{n}\big(s_j - P_r(\ell, b_j)\big)\frac{\partial}{\partial x}P_r(\ell, b_j),$$
$$\frac{\partial}{\partial y} f(\ell) = (-2)\sum_{j=1}^{n}\big(s_j - P_r(\ell, b_j)\big)\frac{\partial}{\partial y}P_r(\ell, b_j),$$

where

$$\begin{aligned} \frac{\partial}{\partial x}P_r(\ell, b_j) &= -10\phi_j \frac{\partial}{\partial x}\log(\|\ell, b_j\|), \\ \frac{\partial}{\partial y}P_r(\ell, b_j) &= -10\phi_j \frac{\partial}{\partial y}\log(\|\ell, b_j\|). \end{aligned} \tag{15}$$

To obtain the Hessian matrix, we have to compute $\frac{\partial^2}{\partial x^2}f(\ell)$, $\frac{\partial^2}{\partial y^2}f(\ell)$, $\frac{\partial^2}{\partial x \partial y}f(\ell)$, and $\frac{\partial^2}{\partial y \partial x}f(\ell)$. For example,

$$\begin{aligned} \frac{\partial^2}{\partial x \partial y} f(\ell) = 2\sum_{j=1}^{n}\Bigg(&\frac{\partial}{\partial x}P_r(\ell, b_j)\frac{\partial}{\partial y}P_r(\ell, b_j) \\ &- (s_j - P_r(\ell, b_j))\frac{\partial^2}{\partial x \partial y}P_r(\ell, b_j)\Bigg), \end{aligned}$$

where

$$\frac{\partial^2}{\partial x \partial y}P_r(\ell, b_j) = -10\phi_j \frac{\partial^2}{\partial x \partial y}\log(\|\ell, b_j\|). \tag{16}$$

The rest of the localization process follows as stated in Section 4.1. Complete steps of Newton-PL are listed in Algorithm 1.

Algorithm 1: Newton-PL Algorithm

```
input : an RSS vector $s$,
        the parameter $\mathbf{x}$ obtained from Eq. (13),
        the locations of base stations
output: an estimated location $\ell_{est}$
1  Select an initial location $\ell^{(0)}$
2  $\alpha \leftarrow 1$
3  $i \leftarrow -1$
4  repeat
5      $i \leftarrow i + 1$
6      Compute the received power $P_r(\ell^{(i)}, b_j)$ at $\ell^{(i)}$
       for each $b_j$ (Eq. (10))
7      Generate the gradient $\nabla f(\ell^{(i)})$ at $\ell^{(i)}$ (Eq. (15))
8      Generate the Hessian matrix $\nabla^2 f(\ell^{(i)})$ at $\ell^{(i)}$
       (Eq. (16))
9      Compute the search direction $d^{(i)}$ (Eq. (8))
10     Jump to the next location $\ell^{(i+1)}$ (Eq. (2))
11 until a termination condition of $\ell^{(i)}$ is met
12 $\ell_{est} \leftarrow \ell^{(i+1)}$
13 return $\ell_{est}$
```

Note that in the positioning phase, no differentiation is involved in (15) and (16) because they are precomputed. After receiving s, the only job we need to do is to plug in s and $\ell^{(i)}$ into (15) and (16). Hence, the online search job can be done efficiently. Furthermore, the search complexity only depends on the number of base stations (i.e., $|\mathcal{B}|$), but is independent of the size of the training data (i.e., $|\mathcal{L}|$). It is expected that $|\mathcal{L}| \gg |\mathcal{B}|$ in most practice.

Fig. 3 illustrates an example using the settings in Fig. 2. A mobile device is located at $\ell^* = (42, 42)$. Initially, $\ell^{(0)} = (35, 12)$. In each iteration, the gradient and Hessian matrix are computed to determine the search direction, as shown by arrows. In this example, Newton-PL only needs four iterations to reach $\ell^{(4)}$, which is very close to ℓ^*. Hence, compared to exhaustively examining $h(\ell_i)$ for all $\ell_i \in \mathcal{L}$, we can expect that the overall computation cost of Newton-PL in the positioning phase will be very limited.

4.3 Newton's Method Using Interpolation (Newton-INT)

The above Newton-PL scheme has an obvious limitation that the path loss is likely unpredictable in indoor environments. Specifically, (9) may not be suitable in an environment with obstacles. Hence, positioning errors will be propagated to (14). To relieve this limitation, Newton-INT adopts the *inverse distance weighted interpolation* (a.k.a. *Shepard interpolation* [34]), a numerical data fitting approach. Similar to Newton-PL, Newton-INT also needs to infer $P_r(\ell, b_j)$ for each b_j from the training data. However, Newton-PL has fixed parameters once (13) is computed in the training phase, but Newton-INT has dynamically changing parameters in its $f(\ell)$ as ℓ changes in the positioning phase. For this reason, the $f(\ell)$ of Newton-INT can continuously pass through $h(\ell_i)$ at each training location ℓ_i, e.g., $f(\ell_i) = \|s, \upsilon_i\|$ for all $\ell_i \in \mathcal{L}$.

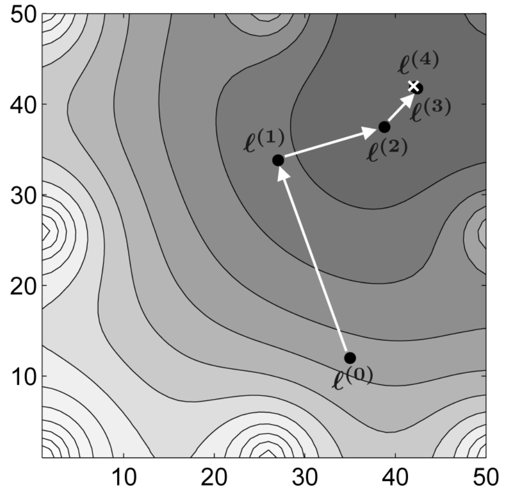

Fig. 3. An example of Newton-PL. Locations $\ell^{(i)}$, $i = 0..4$, are points examined by the discriminant minimization search. Arrows indicate search directions (the last iteration is omitted). The minimizer ℓ^* is shown by a cross.

The main idea is to use a weighted function to predict the signal strength at each $\ell \in \mathcal{F}$. We define the expected RSS of b_j at ℓ as

$$P_r(\ell, b_j) = \frac{1}{\sum_{\ell_i \in \mathcal{L}} w_i} \cdot \sum_{\ell_i \in \mathcal{L}} (w_i \times \upsilon_{i,j}), \tag{17}$$

where $w_i = \|\ell, \ell_i\|^{-\lambda}$. The term $1/\sum_{\ell_i \in \mathcal{L}} w_i$ is for normalization purpose and λ is a system parameter (typically, $\lambda > 0$). In our experiments, we set λ to a very small value, say 0.01, because the adopted Shepard interpolation may generate some suddenly changing gradients in areas with dense training locations, resulting in misleading search directions. With a smaller λ, this can be effectively avoided. Also, (17) guarantees continuity and differentiability properties [34]. Note that when $\ell \to \ell_i$, we have $w_i \to \infty$, leading to $P_r(\ell, b_j) = \upsilon_{i,j}$.

However, there are two concerns in (17). First, it has to refer to all training data to compute $P_r(\ell, b_j)$, thus causing high computation cost. To relax this, we can restrict $P_r(\ell, b_j)$ to be influenced only by a small group $N_r(\ell, \tau)$, which is defined as the set of the first τ training locations in $\mathcal{L}$ nearest to ℓ, where τ is a system parameter. Intuitively, we will not refer to those feature vectors whose training locations are too far from ℓ. In practice, we set $\tau = 2 \sim 5$. In our implementation, we let $\tau = 2$ due to the performance consideration. Given any ℓ, how to quickly identify $N_r(\ell, \tau)$ can be efficiently solved by spatial indexing, such as R-tree [35], which can perform a multivalue key search with a few comparisons by a bounding box concept. For example, in Fig. 4, we can use a bounding box $Box(\ell)$ centering at ℓ to search nearby training locations. The queried training locations are sorted according to their distances to ℓ. Such a search is quite efficient because it only involves 2D spatial queries and the number of the queried training locations are very limited. The search time, however, still depends on the number of training locations.

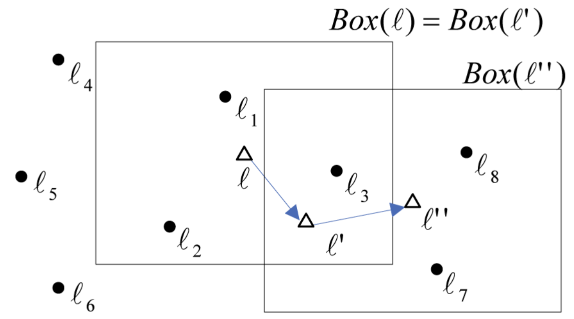

Fig. 4. Reuse of bounding boxes. A query location is marked by a triangle (△) and a training location is marked by a dot (●).

Hence, we have to conduct this operation as less frequently as possible. Toward this goal, several literatures have proposed efficient ways to search nearest neighbors when queries are proceeded in a continuous manner [36], [37], [38]. Here, we propose a simple strategy. First, we can reuse the bounding box $Box(\ell)$ as much as possible. In the example of Fig. 4, for a new spatial query at ℓ', we can reuse the training locations in the previous bounding box $Box(\ell)$ if $\ell' \in Box(\ell)$. However, in the next query, since $\ell'' \notin Box(\ell)$, a new bounding box $Box(\ell'')$ needs to be found. Second, we can adaptively change the size of a bounding box according to its contents. If it covers more than 2τ training locations, we will shrink it in the next iteration; on the other hand, we will expand it if it only covers less than τ training locations.

The second concern in (17) is that the gradient $\nabla P_r(\ell_i, b_j)$ at each training location ℓ_i will be zero [39]. This flatness property will result in unexpected termination of the gradient descent search. To solve this problem, we can guess an artificial gradient $\nabla P_r(\ell_i, b_j) = [\mathcal{G}^x_{i,j}, \mathcal{G}^y_{i,j}]^T$ at each training location ℓ_i for b_j. Several literatures [34], [39] have addressed this issue. We will build a tangent plane at each training location $\ell_i = (x_i, y_i)$ as in [34]:

$$\mathcal{T}_{i,j}(\ell) = \upsilon_{i,j} + \mathcal{G}^x_{i,j}(x - x_i) + \mathcal{G}^y_{i,j}(y - y_i). \tag{18}$$

This plane passes $\upsilon_{i,j}$, that is, $\mathcal{T}_{i,j}(\ell_i) = \upsilon_{i,j}$. To compute $\mathcal{G}^x_{i,j}$ and $\mathcal{G}^y_{i,j}$, we first identify a set $N_g(\ell_i) \subseteq \mathcal{L}$ of training locations which are within a distance of ϵ from ℓ_i (e.g., ϵ can be 10 m). For each $\ell_e = (x_e, y_e) \in N_g(\ell_i)$, we construct an equation:

$$\mathcal{G}^x_{i,j}(x_e - x_i) + \mathcal{G}^y_{i,j}(y_e - y_i) = \upsilon_{e,j} - \upsilon_{i,j}. \tag{19}$$

That is, we expect $\mathcal{T}_{i,j}(\ell)$ to be as closed to $\upsilon_{e,j}$ as possible. Combining these $\kappa = |N_g(\ell_i)|$ equations leads to

$$\underbrace{\begin{bmatrix} x_1 - x_i & y_1 - y_i \\ \vdots & \vdots \\ x_\kappa - x_i & y_\kappa - y_i \end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix} \mathcal{G}^x_{i,j} \\ \mathcal{G}^y_{i,j} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \upsilon_{1,j} - \upsilon_{i,j} \\ \vdots \\ \upsilon_{\kappa,j} - \upsilon_{i,j} \end{bmatrix}}_{C}. \tag{20}$$

By the least-squares analysis,

$$\mathbf{x} = \nabla P_r(\ell_i, b_j) = (A^T A)^{-1} A^T C. \tag{21}$$

Finally, we store these derived gradients in a set

$$\mathcal{T} = \{(\ell_i, \upsilon_{i,j}, \nabla P_r(\ell_i, b_j)) \mid \ell_i \in \mathcal{L}, b_j \in \mathcal{B}\}. \tag{22}$$

With the above modifications, we redefine (17) as

$$P_r(\ell, b_j) = \frac{1}{\sum_{\ell_i \in N_r(\ell,\tau)} w_i} \cdot \sum_{\ell_i \in N_r(\ell,\tau)} \big(w_i \times \mathcal{T}_{i,j}(\ell)\big), \qquad (23)$$

where $P_r(\ell, b_j)$ is a function to which we can plug in any value of ℓ. To compute $P_r(\ell, b_j)$ quickly, we need to maintain $\mathcal{T}$ in a spatial indexing after the training phase. Then, in the positioning phase, given $\boldsymbol{s}$, we can define a discriminant function similar to (14), except that now $P_r(\ell, b_j)$ is as defined in (23). Following this definition, we can derive

$$\frac{\partial}{\partial x} f(\ell) = (-2)\sum_{j=1}^{n}\big(s_j - P_r(\ell, b_j)\big)\frac{\partial}{\partial x}P_r(\ell, b_j),$$
$$\frac{\partial}{\partial y} f(\ell) = (-2)\sum_{j=1}^{n}\big(s_j - P_r(\ell, b_j)\big)\frac{\partial}{\partial y}P_r(\ell, b_j),$$

where

$$\frac{\partial}{\partial x}P_r(\ell, b_j) = \frac{D^X_{j,1}D^X_{j,2} - D^X_{j,3}D^X_{j,4}}{D^X_{j,1}D^X_{j,1}},$$
$$\frac{\partial}{\partial y}P_r(\ell, b_j) = \frac{D^Y_{j,1}D^Y_{j,2} - D^Y_{j,3}D^Y_{j,4}}{D^Y_{j,1}D^Y_{j,1}},$$

and

$$\begin{aligned}
D^X_{j,1} = D^Y_{j,1} &= \sum_{\ell_i \in N_r(\ell,\tau)} w_i,\\
D^X_{j,2} &= \sum_{\ell_i \in N_r(\ell,\tau)} \left(w_i \times \mathcal{G}^x_{i,j} + \mathcal{T}_{i,j}(\ell)\frac{\partial}{\partial x}w_i\right),\\
D^Y_{j,2} &= \sum_{\ell_i \in N_r(\ell,\tau)} \left(w_i \times \mathcal{G}^y_{i,j} + \mathcal{T}_{i,j}(\ell)\frac{\partial}{\partial y}w_i\right),\\
D^X_{j,3} = D^Y_{j,3} &= \sum_{\ell_i \in N_r(\ell,\tau)} \mathcal{T}_{i,j}(\ell)w_i,\\
D^X_{j,4} &= \sum_{\ell_i \in N_r(\ell,\tau)} \frac{\partial}{\partial x}w_i,\\
D^Y_{j,4} &= \sum_{\ell_i \in N_r(\ell,\tau)} \frac{\partial}{\partial y}w_i.
\end{aligned} \qquad (24)$$

Similarly, we can derive the second derivative

$$\frac{\partial^2}{\partial x \partial y} f(\ell) = 2\sum_{j=1}^{n}\left(\frac{\partial}{\partial x}P_r(\ell, b_j)\frac{\partial}{\partial y}P_r(\ell, b_j) - (s_j - P_r(\ell, b_j))\frac{\partial^2}{\partial x \partial y}P_r(\ell, b_j)\right),$$

where

$$\begin{aligned}
\frac{\partial^2}{\partial x \partial y}P_r(\ell, b_j) &= \frac{\partial}{\partial y}\frac{D^X_{j,1}D^X_{j,2} - D^X_{j,3}D^X_{j,4}}{D^X_{j,1}D^X_{j,1}}\\
&= \frac{D^{XY}_{j,1}D^X_{j,2} + D^X_{j,1}D^{XY}_{j,2} - D^{XY}_{j,3}D^X_{j,4} - D^X_{j,3}D^{XY}_{j,4}}{D^X_{j,1}D^X_{j,1}}\\
&\quad - 2\frac{D^{XY}_{j,1}}{D^X_{j,1}}\frac{\partial}{\partial x}P_r(\ell, b_j),
\end{aligned}$$

and

$$\begin{aligned}
D^{XY}_{j,1} &= \sum_{\ell_i \in N_r(\ell,\tau)} \frac{\partial}{\partial y}w_i,\\
D^{XY}_{j,2} &= \sum_{\ell_i \in N_r(\ell,\tau)} \mathcal{G}^x_{i,j}\frac{\partial}{\partial y}w_i + \mathcal{G}^y_{i,j}\frac{\partial}{\partial x}w_i + \mathcal{T}_{i,j}(\ell)\frac{\partial^2}{\partial x \partial y}w_i,\\
D^{XY}_{j,2} &= \sum_{\ell_i \in N_r(\ell,\tau)} \mathcal{G}^x_{i,j}\frac{\partial}{\partial y}w_i + \mathcal{G}^y_{i,j}\frac{\partial}{\partial x}w_i + \mathcal{T}_{i,j}(\ell)\frac{\partial^2}{\partial x \partial y}w_i,\\
D^{XY}_{j,3} &= \sum_{\ell_i \in N_r(\ell,\tau)} w_i \times \mathcal{G}^y_{i,j} + \mathcal{T}_{i,j}(\ell)\frac{\partial}{\partial y}w_i,\\
D^{XY}_{j,4} &= \sum_{\ell_i \in N_r(\ell,\tau)} \frac{\partial^2}{\partial x \partial y}w_i.
\end{aligned} \qquad (25)$$

The other second derivatives $\frac{\partial^2}{\partial x^2}f(\ell)$, $\frac{\partial^2}{\partial y^2}f(\ell)$, and $\frac{\partial^2}{\partial y \partial x}f(\ell)$ can be obtained in the same manner.

Complete steps of Newton-INT are listed in Algorithm 2. Line 6 searches the spatial indexing R for those training locations nearest to a given location. These training locations will be used to compute the following gradient and Hessian matrix. Different from Newton-PL, Newton-INT's discriminant function is dynamically changed according to the referenced training locations. This makes the computation overhead of Newton-INT higher than that of Newton-PL. However, Newton-INT is more suitable for indoor environments.

Algorithm 2: Newton-INT Algorithm

```
input : an RSS vector $\boldsymbol{s}$,
        a spatial indexing $R$ containing $\mathcal{T}$
output: an estimated location $\ell_{est}$
1  Select an initial location $\ell^{(0)}$
2  $\alpha \leftarrow 1$
3  $i \leftarrow -1$
4  repeat
5      $i \leftarrow i + 1$
6      Find the nearest neighborhood $N_r(\ell^{(i)}, \tau)$ from $R$
7      Compute the received power $P_r(\ell^{(i)}, b_j)$ at $\ell^{(i)}$
       for each $b_j$ (Eq. (23))
8      Generate the gradient $\nabla f(\ell^{(i)})$ at $\ell^{(i)}$ (Eq. (24))
9      Generate the Hessian matrix $\nabla^2 f(\ell^{(i)})$ at $\ell^{(i)}$
       (Eq. (25))
10     Compute the search direction $d^{(i)}$ (Eq. (8))
11     Jump to the next location $\ell^{(i+1)}$ (Eq. (2))
12 until a termination conditions of $\ell^{(i)}$ is met
13 $\ell_{est} \leftarrow \ell^{(i+1)}$
14 return $\ell_{est}$
```

In some cases, the selected set $N_r(\ell, \tau)$ may not have the RSS patterns for a specific base station b_j. That will make calculating (24) and (25) impossible. To solve this problem, we can just skip b_j. Alternatively, we may increase τ. The third option is to adopt the path loss model in this situation. We can build the path loss model for each base station as stated in Section 4.2. If the RSS patterns of $N_r(\ell, \tau)$ contain b_j, we apply (24) and (25); otherwise, we apply (15) and (16). These strategies can make Newton-INT more practicable.

TABLE 1
Comparison of Time and Space Complexities

	Time Complexity	Space Complexity
NNSS	$O(C_{PM} \times \lvert\mathcal{L}\rvert)$	$O(\lvert\mathcal{L}\rvert \times \lvert\mathcal{B}\rvert)$
k-means	$O(C_{PM} \times (k + \frac{\lvert\mathcal{L}\rvert}{k}) + C_{DL})$	$O((k + \frac{\lvert\mathcal{L}\rvert}{k}) \times \lvert\mathcal{B}\rvert)$
Newton-PL	$O(C_{PL} \times i_{max})$	$O(\lvert\mathcal{B}\rvert)$
Newton-INT	$O((C_{INT} + C_{DL}) \times i_{max})$	$O(\tau \times \lvert\mathcal{B}\rvert)$

4.4 Discussion

The DMS-based localization has several advantages over traditional exhaustive-search-based schemes. First, it can achieve fine-grained positioning as long as the corresponding discriminant functions are continuous and differentiable. This also reduces the calibration labor cost due to collecting less training data. Second, as to be shown in our simulation studies, our algorithms normally run much faster than exhaustive schemes, especially in large environments. Table 1 compares the time complexities of NNSS [5], k-means clustering [11], Newton-PL, and Newton-INT, where C_{PM} is the cost to compare two RSS vectors, and C_{PL} and C_{INT} are the costs to perform one iteration of Newton-PL and Newton-INT search, respectively. For a simple RSS comparison, like (1), the complexity is only dependent on the number of base stations, i.e., $C_{PM} = O(|\mathcal{B}|)$. Newton-PL's search cost C_{PL} has the same complexity as C_{PM}, but the iteration bound i_{max} is usually much less than $|\mathcal{L}|$ or $k+\frac{|\mathcal{L}|}{k}$. Hence, the overall time complexity of Newton-PL is less than NNSS and k-means algorithms. As for Newton-INT, its search cost C_{INT} does depend on $|\mathcal{L}|$ because its spatial indexing contributes a logarithmic complexity, i.e., $C_{INT} = O(\log|\mathcal{L}| + \tau|\mathcal{B}|)$. However, in practice, it would perform better than the NNSS and k-means algorithms as $|\mathcal{L}|$ becomes larger. Note that k-means and Newton-INT may involve a dynamic memory loading cost C_{DL} to load training data from database when memory space is limited. If memory space is not essential, loading all data into memory would be more efficient, but the space requirement will become $O(|\mathcal{L}| \times |\mathcal{B}|)$.

The third advantage is on space complexity. As shown in Table 1, the storage requirements of Newton-PL and Newton-INT are much lower than those of NNSS and k-means because in most practical applications, we have $|\mathcal{L}| \gg |\mathcal{B}|$. Interestingly, for Newton-PL, it does not need the training database during the positioning phase, because it has been transformed to the parameters of the discriminant function. For Newton-INT, only those neighboring training data are needed during the positioning phase. This would be beneficial to enable positioning by portable devices themselves.

However, our DMS-based solution has an inherent limitation, i.e., the local minimum problem. According to our simulation studies in Section 5.1.4, falling into a local minimum is not rare if the initial point is not selected carefully, especially in a complex environment. Several literatures have discussed this issue [40], [41]. Fortunately, for the localization problem, the local minimum problem can easily be avoided because in most LBSs, continuous locations are needed, which means that the previous location estimation can be used as the initial point of the next gradient search.

5 Simulation and Experiment Results

5.1 Simulation Results

We consider a field of size $250 \times 250\ \mathrm{m}^2$ with 169 base stations placed at $(20 \times i, 20 \times j)$, $i = 0..12$ and $j = 0..12$, and 2 m above the ground. All of these base stations will emit signals used for the localization purpose. To complicate the situation, some vertical and horizontal walls are given along line segments $((20 \times i, 0), (20 \times i, 250))$ and $((0, 20 \times i), (250, 20 \times i))$, $i = 0..12$. Training locations are at integer grid points, where the grain size is a parameter. At each training location, 50 training data are collected and their average is taken. We modify the RIM [24] to model RSSs:

$$P_r(\ell, b_j) = P_t^{VSP}(b_j) - PL^{DOI}(\ell, b_j) - PL^{WAF}(\ell, b_j) + N(0, \sigma), \tag{26}$$

where $P_t^{VSP}(b_j)$ is to model the transmit power (which may vary among different hardware), $PL^{DOI}(\ell, b_j)$ is to model the path loss (which has nonisotropic and continuous properties), $PL^{WAF}(\ell, b_j)$ is to simulate the signal attenuation caused by obstacles, and $N(0, \sigma)$ is a zero-mean normal random variable with a standard deviation σ to represent dynamic noise. Signal sensitivity is bounded by s_{min}, i.e., any $P_r(\ell, b_j)$ less than s_{min} is ignored.

In RIM, `VSP` stands for *variance of sending power* and is formulated by

$$P_t^{VSP}(b_j) = P_t \times (1 + N(0, \texttt{VSP})), \tag{27}$$

where P_t is a constant transmit power and $N(0, \texttt{VSP})$ is a zero-mean normal random variable with a standard deviation `VSP`. In the beginning of a simulation, each base station b_j randomly selects its $P_t^{VSP}(b_j)$ as its transmit power. `DOI` stands for *degree of irregularity* to control the amount of path loss in different directions and is formulated by

$$PL^{DOI}(\ell, b_j) = PL(\|\ell, b_j\|) \times K_i, \tag{28}$$

where $PL(\|\ell, b_j\|)$ is the obstacle-free path loss equal to (9) and K_i is to model the level of irregularity at degree i ($i = 0..359$) such that

$$K_i = \begin{cases} 1, & \text{if } i = 0, \\ K_{i-1} \pm W(0, \beta, \eta) \times \texttt{DOI}, & \text{if } i = 1..359, \end{cases} \tag{29}$$

where $|K_0 - K_{359}| \leq \texttt{DOI}$ and $W(0, \beta, \eta)$ is a zero-mean Weibull random variable with a slope parameter β and a scale parameter η. Here, we let $\beta = 1$ and $\eta = 0.1$. To model the complicated partitions in an environment and the related signal attenuation, `WAF` stands for *wall attenuation factor* and is formulated by [5]:

$$PL^{WAF}(\ell, b_j) = \min(N_{obs}, N_{max}) \times \texttt{WAF}, \tag{30}$$

where N_{obs} is the number of obstacles (walls) on the line-of-sight path from b_j to ℓ, N_{max} is the maximum number of obstacles which can influence $PL^{WAF}(\ell, b_j)$, and `WAF` is the amount of signal attenuation caused by one wall.

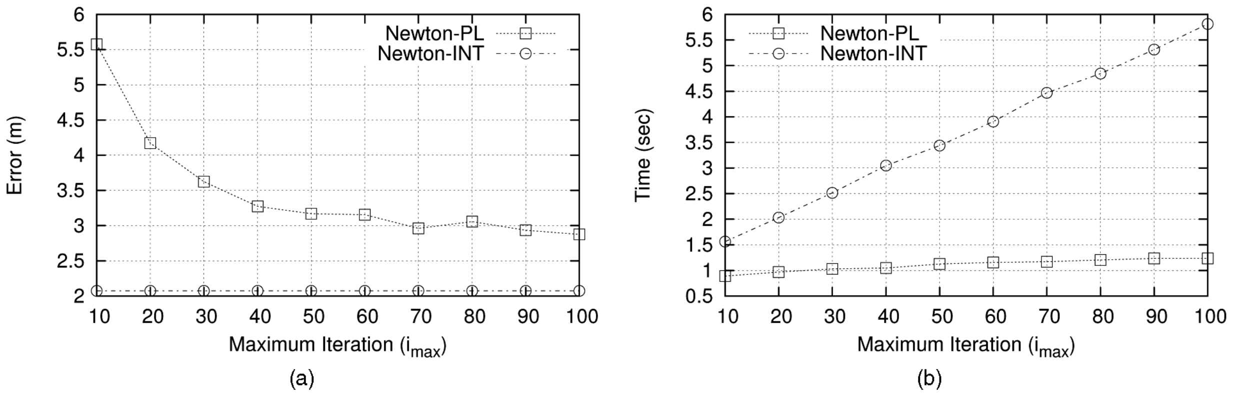

Fig. 5. Comparisons of (a) positioning error and (b) processing time for Newton-PL and Newton-INT under different i_{max} values.

A mobile device moving by a random waypoint model is simulated. It switches between *moving* and *pausing* states. In the moving state, it uniformly selects a destination in $\mathcal{F}$ and moves to it at 1 m/sec. After reaching the destination, it switches to the pausing state and stays there for 3 sec. The mobile device measures RSS from all base stations every 1 sec. The total simulation time is 10,000 sec. Note that for simplicity, we assume that the mobile device can walk through the above obstacles. (Refer to [42] for mobility models which can route around obstacles.) The default simulation parameters are $P_t = 15$ dBm, $d_0 = 1$ m, $PL(d_0) = 37.3$ dBm, $\phi = 4$, $s_{min} = -96$ dBm, $\sigma = 3$, $\texttt{VSP} = 0.2$, $\texttt{DOI} = 0.01$, $\texttt{WAF} = 3$, $N_{max} = 4$, $\Delta\ell_{min} = 1$ m, $\tau = 2$, $\lambda = 0.01$, $\epsilon = 10$ m, and training grain size = 5 m.

We compare our algorithms against NNSS with a spatial filtering mechanism (denoted as SF-NNSS) and the k-means clustering algorithm (denoted as k-MEANS) [11], [21]. In SF-NNSS, it only considers those ℓ_i whose v_i contains at least $|s| - 1$ base stations in s, where $|s|$ denotes the number of the actually received base stations. If all training locations are filtered out, we will loosen the condition and consider ℓ_i containing at least $|s| - 2$ base stations in s and so on. The k-MEANS algorithm is a well-known technique to separate a set of data into k subsets. It is an iterative process. Each iteration has a regrouping step and a cluster centroid computation step. Initially, k cluster centroids are randomly selected from the feature set $\mathcal{V}$. Then, in the regrouping step, each training location ℓ_i is grouped to the nearest cluster by measuring the distance between v_i and each centroid. After regrouping, the centroid of each new-born cluster is calculated by averaging the feature vectors of all training locations in this cluster. This process will be repeated until convergence is reached. During the positioning phase, only the training locations in the cluster whose centroid is closest to s are compared to s. Its computation cost depends on two factors: $|\mathcal{L}|$ and k. In our simulation, we set k around $\sqrt{|\mathcal{L}|}$.

Two performance metrics are considered: positioning error and processing time. The processing time is measured by the actual CPU cycles consumed from the first to the last positioning tasks (i.e., 10,000 rounds). In our simulation, we adopt a PC with an Intel Core 2 Qual (Q6600 at 2.40 GHz).

5.1.1 Study of Maximum Iteration

For Newton-PL and Newton-INT algorithms, i_{max} indicates the largest number of iterations that the algorithms can run. Fig. 5 compares Newton-PL and Newton-INT under different i_{max} values. Fig. 5a shows that Newton-INT is quite insensitive to i_{max} in terms of positioning error. So, a small i_{max} is sufficient. Newton-PL reaches a stable error after $i_{max} \geq 50$. Fig. 5b shows the incurred processing time. We observe that Newton-INT has a steeper increasing trend than Newton-PL. This is because C_{INT} is much larger than C_{PL}. So, we can allow a larger i_{max} for Newton-PL to converge. Considering both Figs. 5a and 5b, we will set i_{max} to 100 and 10 for Newton-PL and Newton-INT, respectively, in the rest of the simulations and experiments.

5.1.2 Influence of Training Grain Size

In Fig. 6, we vary the density of training locations. A finer grain size incurs longer processing time for both SF-NNSS and k-MEANS. However, changing grain sizes does not have dramatic influence on Newton-PL. As for Newton-INT, its processing time slightly increases as the grain size decreases. So, the proposed spatial indexing and search strategy work quite well. In general, $|\mathcal{L}|$ should be quite large, making our DMS-based algorithms more attractive in terms of computation cost.

As to positioning error, SF-NNSS, k-MEANS, and Newton-INT all benefit from finer grain sizes because they can better capture the signal fading trend as more training data are given. However, without sufficient training data, SF-NNSS and k-MEANS will be much worse than our algorithms due to low resolution of training locations. Among all, Newton-INT performs the best in most cases because it can predict signal patterns quite well.

5.1.3 Parameters of the Radio Model

Parameter DOI is to control the signal irregularity. As shown in Fig. 7, DOI has little impact on processing time for all algorithms. However, a larger DOI incurs a much larger positioning error on Newton-PL because it makes a strong assumption that the decay of signals follows a logarithmic function, which usually does not hold in indoor environments. On the other hand, SF-NNSS, k-MEANS, and Newton-INT are all quite insensitive to

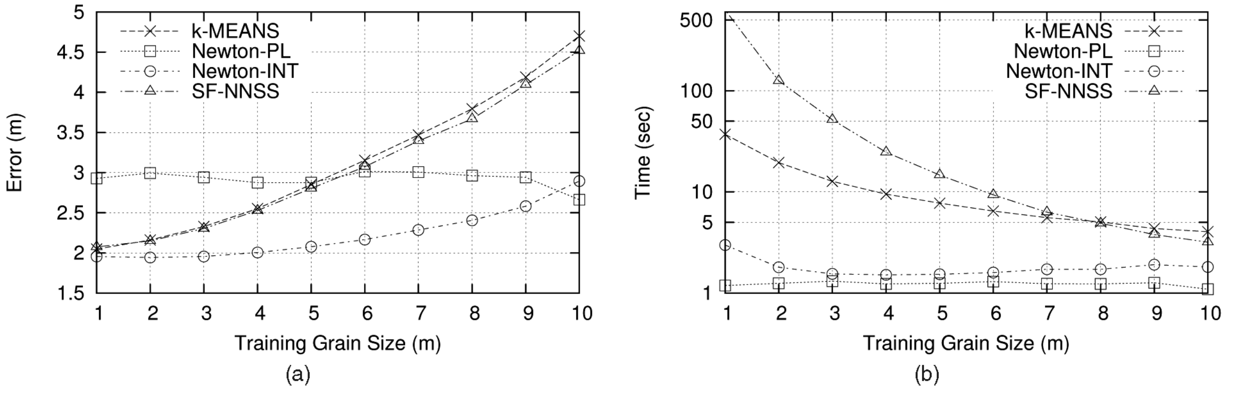

Fig. 6. Comparisons of (a) positioning error and (b) processing time under different training grain sizes.

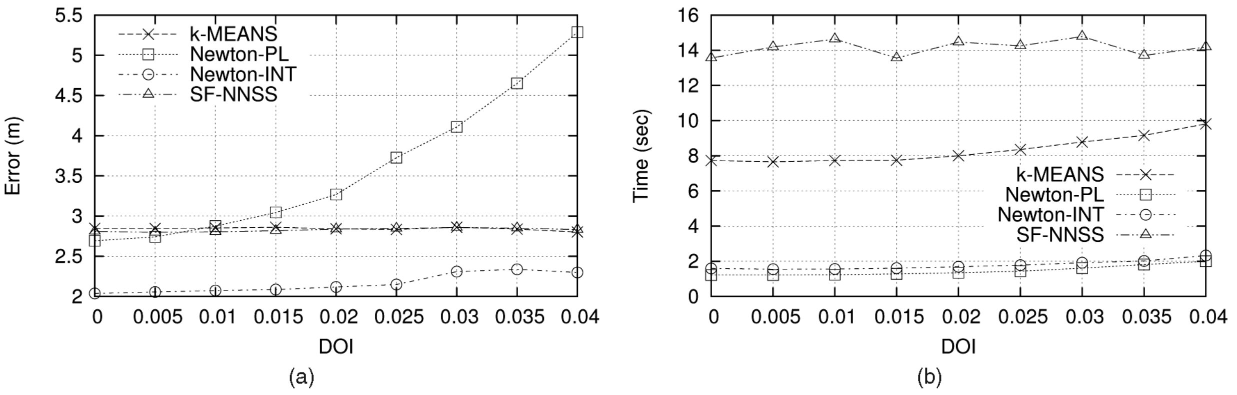

Fig. 7. Comparisons of (a) positioning error and (b) processing time under different DOI values.

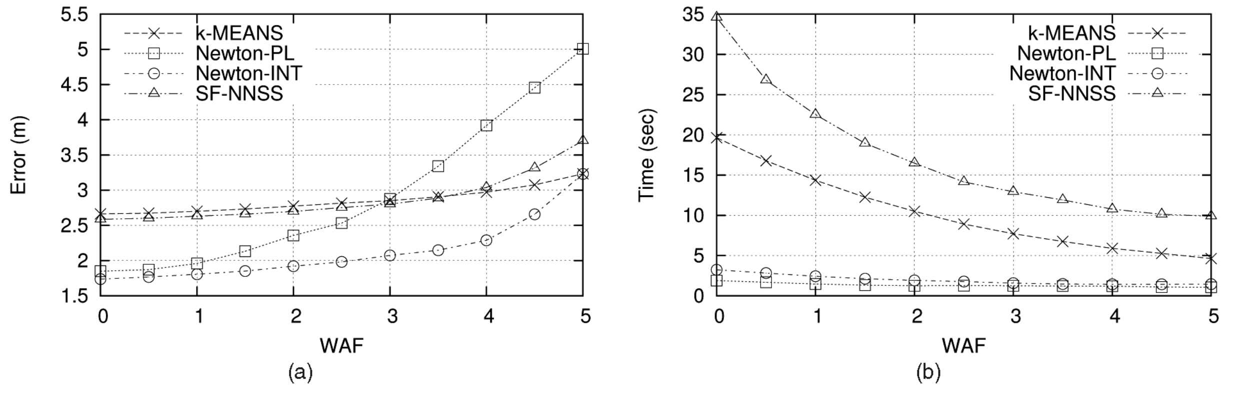

Fig. 8. Comparisons of (a) positioning error and (b) processing time under different WAF values.

DOI in terms of positioning error. In all cases, Newton-INT performs the best.

Different from DOI, parameter WAF is to model the impact of obstacles on signals. We can expect a dramatic change in our discriminant function values whenever a wall is encountered. These steeps increase the difficulty of our search job because the derived gradients might be unsuitable. As shown in Fig. 8a, a larger WAF is a disadvantage to all algorithms. But Newton-PL is most sensitive to this change. Interestingly, as compared to Fig. 7a, sudden changes of signals (WAF) trouble Newton-INT more than continuous changes (DOI). On the contrary, WAF impacts SF-NNSS and k-MEANS from a different angle. A larger WAF reduces the number of base stations that a device can see. This makes training locations less distinguishable, thus increasing the positioning error. This also reduces computation costs for SF-NNSS and k-MEANS, as shown in Fig. 8b. Contrarily, the processing time of Newton-PL and Newton-INT remains quite flat as WAF increases because less distinguishable environments also induce more search iterations.

Parameter VSP is to model the variance of transmission powers among base stations due to hardware differences. However, for pattern-matching localization, this factor contributes little influence because we do not make any assumption about hardware consistency. Fig. 9 verifies this fact.

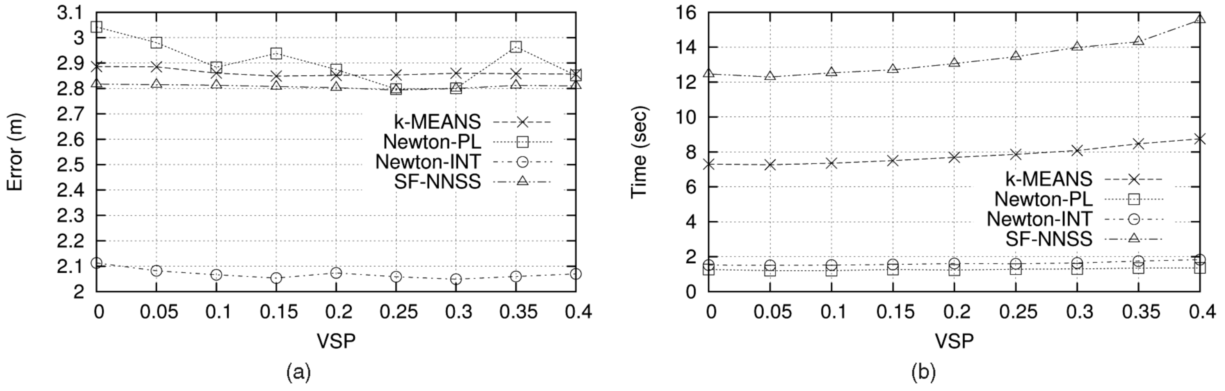

Fig. 9. Comparisons of (a) positioning error and (b) processing time under different VSP values.

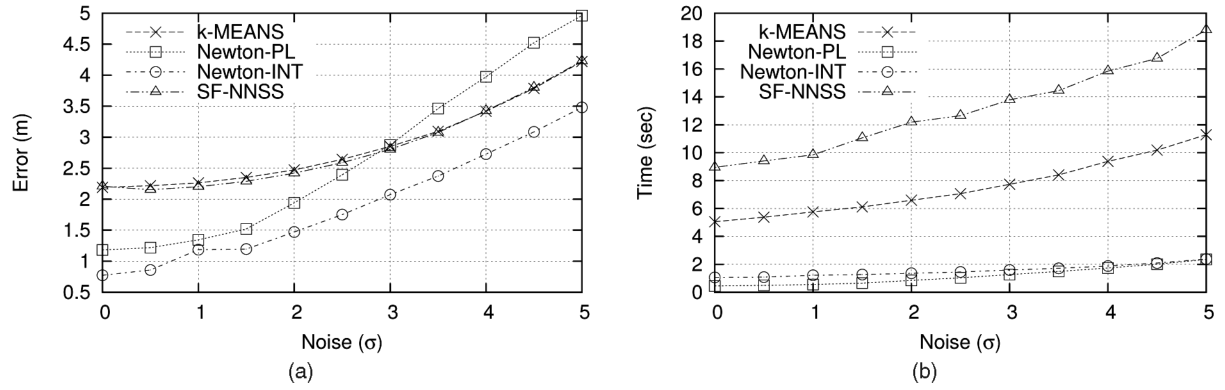

Fig. 10. Comparisons of (a) positioning error and (b) processing time under different noise levels.

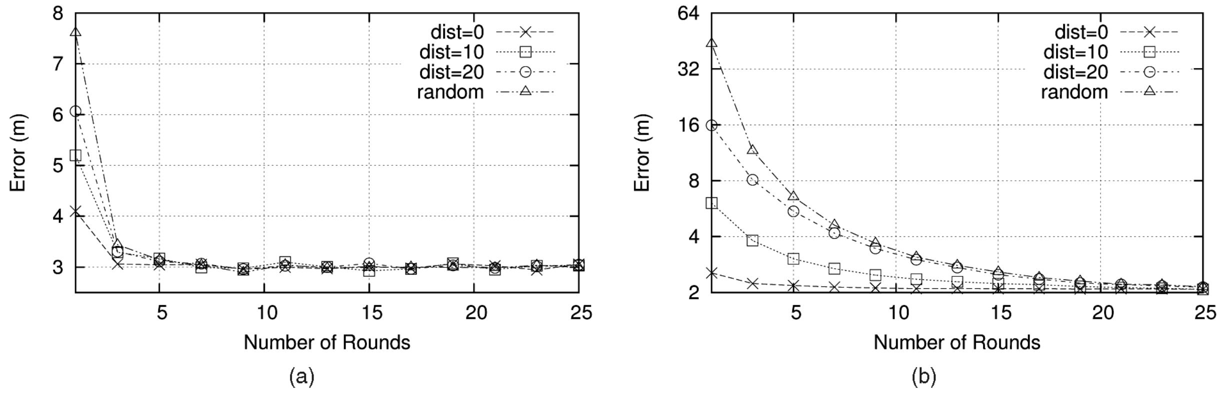

Fig. 11. Simulation studies of convergence rate of (a) Newton-PL and (b) Newton-INT when the initial location of the first round is randomly selected.

In Fig. 10, we conduct evaluations under different noise levels. Fig. 10a shows that all algorithms suffer from higher noise levels. SF-NNSS and k-MEANS are constrained by the grain size, so even with low noise levels, their positioning accuracy cannot be improved much. However, when $\sigma > 3$, the noise level becomes the major factor of positioning errors.

5.1.4 Influence of Initial Point Selection

Here, we select a random initial point for our DMS-based algorithms and run multiple "rounds" of localization. The first few rounds may fall into local minimum areas, and we are interested in observing how continuous rounds may relieve this problem (each round uses the positioning result of the previous round as its initial point). In our simulation, the initial point of the first round is set to $(x^* + d \times \cos\theta, y^* + d \times \sin\theta)$, where (x^*, y^*) is the actual initial location, θ is randomly distributed in $[0, 360)$, and d is a normal random variable $N(dist, 5)$. Here, a larger $dist$ means less knowledge about the initial location of the target. Fig. 11 illustrates the average error distances from the 1st to the 25th rounds for $dist = 0$, $dist = 10$, $dist = 20$, and "$random$" (in "$random$," the initial point of the first round is randomly selected from $\mathcal{F}$).[2] We observe that Newton-PL converges

2. Note that each value in Fig. 11 represents the average of multiple simulations, including those encountering and not encountering the local minimum problem. However, accurately distinguish these two situations by programs is difficult, if not impossible.

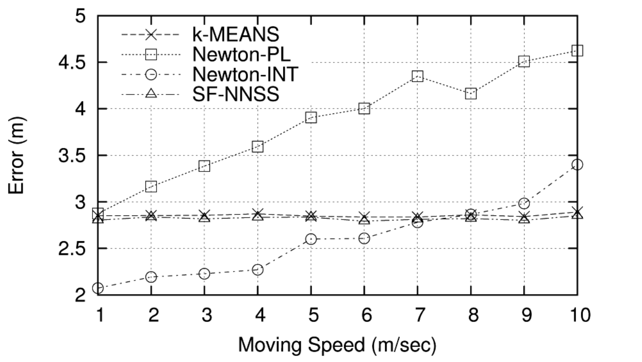

Fig. 12. Simulation studies of positioning error under different moving speeds.

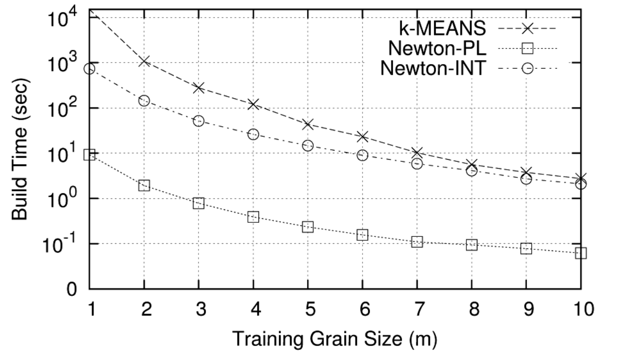

Fig. 13. Simulation studies of model building time under different training grain sizes.

quicker than Newton-INT. However, after entering a steady state, Newton-INT has better accuracy than Newton-PL. This is because the discriminant function of Newton-PL can better express a global trend to find where the actual location is, but is weaker in expressing the detailed signal patterns in small regions.

Although the initial point of the current round can be set to the estimated location of the previous round, the user's moving speed may worsen the problem. When the speed is high, the adopted initial point may already deviated a lot from the user's current location. From Fig. 12, we see that Newton-PL and Newton-INT both suffer from higher moving speeds. However, this is not the case for SF-NNSS and k-MEANS. Fortunately, moving speed over 5 m/sec is already quite unlikely for pedestrians.

5.1.5 Model Building Time

Here, we study the (offline) cost to build a localization model. Rebuilding the model is needed when the environments or the locations of base stations change. SF-NNSS takes a lazy strategy, so it has no building cost (except collecting training data). Fig. 13 compares the model building time under different training grain sizes. Newton-PL can calculate its path loss model very quickly, so it has the lowest cost. Newton-INT is about two order more costly than Newton-PL, but is about one order less costly than k-MEANS. Although both Newton-INT and k-MEANS are more costly than Newton-PL, they have a difference. The k-MEANS algorithm is an iterative process. When an environment

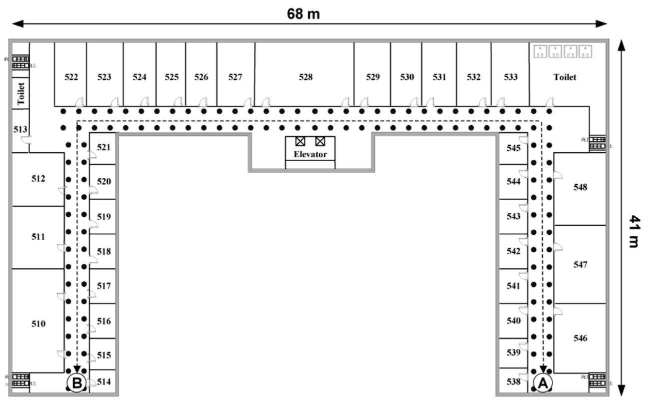

Fig. 14. Our experiment environment at the Computer Science Building, National Chiao Tung University. Training locations are labeled by dots (•). Testing data are collected along the dotted line between Ⓐ and Ⓑ.

changes, it has to repeat the iterative process again. However, the spatial indexing in Newton-INT allows us to add or delete training data easily.

5.2 Experiment Results

Next, we further verify our results in a median-scale real environment, as show in Fig. 14. Training data are collected through Atheros AR5007EG NICs from 124 training locations, each separated by 2 m, in the public corridor. At each training location, 50 samples are collected at a sampling rate of 7-9 samples per second. Since this building is heavily deployed with WiFi APs, each sample contains about 30 base stations in average. Overall, we discover 131 base stations. However, locations of these base stations cannot be precisely identified since this environment is not strictly controlled. We cannot even identify their hardware specifications. We also collect data at 117 testing locations, each separated by 1 m, for testing purpose (along the dotted line in Fig. 14). In our experiment, we evaluate a user walking back-and-forth between the two ends (Ⓐ and Ⓑ) of the corridor. Whenever a testing location is traversed, one random pattern from that location is picked. This brings some randomness to walking traces. Below, we only measure the positioning accuracy. The processing time is relatively underrepresented because the testing environment is not large.

Fig. 15 draws positioning error against the amount of training data (we randomly pick some percentage of the 124 training locations for experiment; note that this also makes the pattern of training locations irregular). Naturally, the error reduces as there are more training locations. Newton-PL performs the worst because correctly predicting path loss is hard. Through real tests, we find that signal quality can easily be interfered by room partitions and moving people. Besides, the locations of base stations are estimated from the given training data, resulting in further errors. This result confirms our intuition that propagational models are not suitable for indoor localization. Overall, Newton-INT has the smallest positioning error (below 2 m even if we pick 10 percent of training locations). The results generally conform to Fig. 6a.

Fig. 16 further tests the impact of moving speed. In Fig. 15, we sequentially choose testing locations between Ⓐ and Ⓑ,

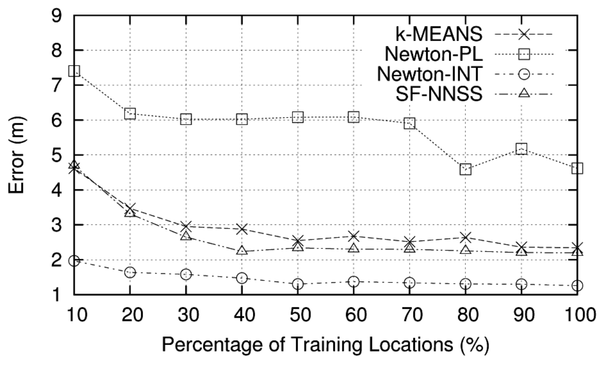

Fig. 15. Experiment results of the positioning error under different amount of training data.

each separated by 1 m. Here, we adopt the same trace, but pick testing locations with some gap. This gives a higher moving speed. Fig. 16 shows the result. Moving speed does not have much impact on positioning error, except Newton-PL. The results generally conform to our simulation results in Fig. 12, except that Newton-INT is quite insensitive to moving speed. One possible reason is that our experiment environment is not as complex as the simulation environment because the traces do not cross walls.

6 Conclusions

In this paper, we have presented a novel DMS-based localization methodology. It is scalable to large environments for three reasons. First, it does not scale or scales up at a slower speed with the increase of training locations. Most time-consuming work can be done offline at the training stage. The online positioning work is quite lightweight. Second, DMS-based localization has higher positioning accuracy due to the continuity property of the discriminant functions. Third, its space complexity for gradient search is quite low because training data can be abstracted by a few scalars of the discriminant function at the offline stage. We believe that these are enabling features of large-scale LBSs [43]. Future research directions may include:

- Development of more efficient discriminant functions.
- Dynamic adjustment of the parameters of discriminant functions when part of the training data is changed.
- Server-based versus client-based localization: traditional fingerprinting localization systems rely on the localization server to conduct the online search job. Since our DMS-based solutions are quite lightweight, the online search job may be conducted by the mobile devices themselves. It deserves to study how to trade the computation with the communication cost. This may even enhance privacy because localization is done locally.
- Integration of the DMS-based solutions with other technologies, such as Kalman filter [44], particle filter [45], averaging method [46], and scrambling method [12], to further improve the accuracy.

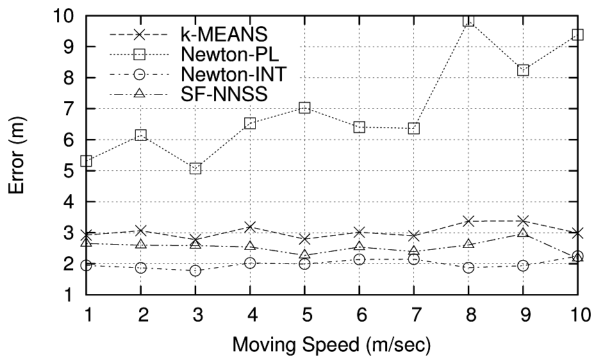

Fig. 16. Experiment results of the positioning error under different moving speeds (percentage of training locations $= 30\%$).

Acknowledgments

Y.-C. Tseng's research is cosponsored by MoE ATU Plan, by NSC grants 97-3114-E-009-001, 97-2221-E-009-142-MY3, 98-2219-E-009-019, and 98-2219-E-009-005, by MOEA 98-EC-17-A-02-S2-0048, and 98-EC-17-A-19-S2-0052, and by ITRI, Taiwan.

References

[1] P. Enge and P. Misra, "Special Issue on Global Positioning System," *Proc. IEEE*, vol. 87, no. 1, pp. 3-15, Jan. 1999.
[2] Skyhook Wi-Fi Positioning System, http://www.skyhookwireless.com, 2010.
[3] Navizon Peer-to-Peer Wireless Positioning, http://www.navizon.com, 2010.
[4] Y. Zhao, "Standardization of Mobile Phone Positioning for 3G Systems," *IEEE Comm. Magazine*, vol. 40, no. 7, pp. 108-116, July 2002.
[5] P. Bahl and V.N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," *Proc. IEEE INFOCOM*, vol. 2, pp. 775-784, 2000.
[6] P. Bahl, A. Balachandran, and V. Padmanabhan, "Enhancements to the RADAR User Location and Tracking System," Technical Report MSR-TR-00-12, Microsoft Research, 2000.
[7] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, "A Probabilistic Approach to WLAN User Location Estimation," *Int'l J. Wireless Information Networks*, vol. 9, no. 3, pp. 155-164, 2002.
[8] V. Seshadri, G.V. Záruba, and M. Huber, "A Bayesian Sampling Approach to In-Door Localization of Wireless Devices Using Received Signal Strength Indication," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm.*, pp. 75-84, 2005.
[9] M. Brunato and R. Battiti, "Statistical Learning Theory for Location Fingerprinting in Wireless LANs," *Computer Networks*, vol. 47, no. 6, pp. 825-845, 2005.
[10] J.J. Pan, J.T. Kwok, Q. Yang, and Y. Chen, "Multidimensional Vector Regression for Accurate and Low-Cost Location Estimation in Pervasive Computing," *IEEE Trans. Knowledge and Data Eng.*, vol. 18, no. 9, pp. 1181-1193, Sept. 2006.
[11] S.-P. Kuo, B.-J. Wu, W.-C. Peng, and Y.-C. Tseng, "Cluster-Enhanced Techniques for Pattern-Matching Localization Systems," *Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems*, 2007.
[12] S.-P. Kuo and Y.-C. Tseng, "A Scrambling Method for Fingerprint Positioning Based on Temporal Diversity and Spatial Dependency," *IEEE Trans. Knowledge and Data Eng.*, vol. 20, no. 5, pp. 678-684, May 2008.
[13] G.V. Záruba, M. Huber, F.A. Kamangar, and I. Chlamtac, "Monte Carlo Sampling Based In-Home Location Tracking with Minimal RF Infrastructure Requirements," *Proc. IEEE Global Telecomm. Conf. (Globecom '04)*, vol. 6, pp. 3624-3629, 2004.
[14] Z. Xiang, H. Zhang, J. Huang, S. Song, and K.C. Almeroth, "A Hidden Environment Model for Constructing Indoor Radio Maps," *Proc. IEEE Int'l Symp. World Wireless Mobile and Multimedia Networks*, pp. 395-400, 2005.

[15] L.F.M. de Moraes and B.A.A. Nunes, "Calibration-Free WLAN Location System Based on Dynamic Mapping of Signal Strength," *Proc. ACM MobiWac*, pp. 92-99, 2006.

[16] X. Chai and Q. Yang, "Reducing the Calibration Effort for Probabilistic Indoor Location Estimation," *IEEE Trans. Mobile Computing*, vol. 6, no. 6, pp. 649-662, June 2007.

[17] P. Krishnan, A.S. Krishnakumar, W.-H. Ju, C. Mallows, and S. Ganu, "A System for LEASE: Location Estimation Assisted by Stationary Emitters for Indoor RF Wireless Networks," *Proc. IEEE INFOCOM*, vol. 2, pp. 1001-1011, 2004.

[18] B. Li, Y. Wang, H.K. Lee, A. Dempster, and C. Rizos, "A New Method for Yielding a Database of Location Fingerprints in WLAN," *Proc. IEE Comm.*, vol. 152, no. 5, pp. 580-586, 2005.

[19] T.-C. Tsai, C.-L. Li, and T.-M. Lin, "Reducing Calibration Effort for WLAN Location and Tracking System using Segment Technique," *Proc. IEEE Int'l Conf. Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 2, pp. 46-51, 2006.

[20] S. Ganu, A.S. Krishnakumar, and P. Krishnan, "Infrastructure-Based Location Estimation in WLAN Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '04)*, vol. 1, pp. 465-470, 2004.

[21] Y. Chen, Q. Yang, J. Yin, and X. Chai, "Power-Efficient Access-Point Selection for Indoor Location Estimation," *IEEE Trans. Knowledge and Data Eng.*, vol. 18, no. 7, pp. 877-888, July 2006.

[22] M. Youssef, A. Agrawala, and U. Shankar, "WLAN Location Determination via Clustering and Probability Distributions," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm.*, pp. 143-150, 2003.

[23] Y. Ji, S. Biaz, S. Pandey, and P. Agrawal, "ARIADNE: A Dynamic Indoor Signal Map Construction and Localization System," *Proc. ACM Int'l Conf. Mobile Systems, Applications, and Services*, pp. 151-164, 2006.

[24] G. Zhou, T. He, S. Krishnamurthy, and J.A. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks," *Proc. ACM Int'l Conf. Mobile Systems, Applications, and Services*, pp. 125-138, 2004.

[25] J. Krumm and J. Platt, "Minimizing Calibration Effort for an Indoor 802.11 Device Location Measurement System," Technical Report MSR-TR-2003-82, Microsoft Research, 2003.

[26] A. Agiwal, P. Khandpur, and H. Saran, "LOCATOR: Location Estimation System for Wireless LANs," *Proc. ACM Int'l Workshop Wireless Mobile Applications and Services WLAN Hotspots*, pp. 102-109, 2004.

[27] A. Kushki, K.N. Plataniotis, and A.N. Venetsanopoulos, "Kernel-Based Positioning in Wireless Local Area Networks," *IEEE Trans. Mobile Computing*, vol. 6, no. 6, pp. 689-705, June 2007.

[28] A. Haeberlen, E. Flannery, A.M. Ladd, A. Rudys, D.S. Wallach, and L.E. Kavraki, "Practical Robust Localization over Large-Scale 802.11 Wireless Networks," *Proc. ACM Int'l Conf. Mobile Computing and Networking*, pp. 70-84, 2004.

[29] Y.-C. Chen, J.-R. Chiang, H. hua Chu, P. Huang, and A.W. Tsui, "Sensor-Assisted Wi-Fi Indoor Location System for Adapting to Environmental Dynamics," *Proc. ACM Int'l Conf. Modeling, Analysis and Simulation Wireless and Mobile Systems*, pp. 118-125, 2005.

[30] E.K. Chong and S.H. Żak, *An Introduction to Optimization*, second ed. John Wiley and Sons, 1995.

[31] J.J. Moré, "The Levenberg-Marquardt Algorithm: Implementation and Theory," *Lecture Notes in Mathematics*, vol. 630, pp. 105-116, Springer, 1977.

[32] T.S. Rappaport, *Wireless Communications: Principles and Practice.* Prentice Hall PTR, 1996.

[33] D. Han, D.G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan, "Access Point Localization Using Local Signal Strength Gradient," *Proc. Int'l Conf. Passive and Active Measurement*, 2009.

[34] D. Shepard, "A Two-Dimensional Interpolation Function for Irregularly-Spaced Data," *Proc. ACM Nat'l Conf.*, pp. 517-524, 1968.

[35] P. Rigaux, M. Scholl, and A. Voisard, *Spatial Databases: With Application to GIS.* Morgan Kaufmann, 2001.

[36] Y. Tao and D. Papadias, "Time-Parameterized Queries in Spatio-Temporal Databases," *Proc. ACM SIGMOD*, pp. 334-345, 2002.

[37] Y. Tao, D. Papadias, and Q. Shen, "Continuous Nearest Neighbor Search," *Proc. Int'l Conf. Very Large Data Bases*, pp. 287-298, 2002.

[38] B. Zheng, W.-C. Lee, and D.L. Lee, "Search Continuous Nearest Neighbors on the Air," *Proc. Int'l ICST Conf. Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 236-245, 2004.

[39] W.J. Gordon and J.A. Wixom, "Shepard's Method of 'Metric Interpolation' to Bivariate and Multivariate Interpolation," *Math. of Computation*, vol. 32, no. 141, pp. 253-264, 1978.

[40] P.R. Chowdhury, Y.P. Singh, and R.A. Chansarkar, "Hybridization of Gradient Descent Algorithms with Dynamic Tunneling Methods for Global Optimization," *IEEE Trans. Systems, Man and Cybernetics, Part A: System and Humans*, vol. 30, no. 3, pp. 384-390, May 2000.

[41] O. Maron and A. Moore, "Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation," *Advances in Neural Information Processing Systems*, vol. 6, pp. 59-66, Morgan Kauffmann, Apr. 1994.

[42] A. Jardosh, E.M. Belding-Royer, K.C. Almeroth, and S. Suri, "Towards Realistic Mobility Models for Mobile Ad Hoc Networks," *Proc. ACM MobiCom*, pp. 217-229, 2003.

[43] S.-P. Kuo, S.-C. Lin, B.-J. Wu, Y.-C. Tseng, and C.-C. Shen, "GeoAds: A Middleware Architecture for Music Service with Location-Aware Advertisement," *Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems*, 2007.

[44] I. Guvenc, C.T. Abdallah, R. Jordan, and O. Dedeoglu, "Enhancements to RSS Based Indoor Tracking Systems Using Kalman Filters," *Proc. GSPx and Int'l Signal Processing Conf.*, pp. 91-102, 2003.

[45] J. Letchner, D. Fox, and A. LaMarca, "Large-Scale Localization from Wireless Signal Strength," *Proc. Nat'l Conf. Artificial Intelligence*, pp. 15-20, 2005.

[46] M. Youssef and A. Agrawala, "The Horus WLAN Location Determination System," *Proc. ACM Int'l Conf. Mobile Systems, Applications, and Services*, pp. 205-218, 2005.

Sheng-Po Kuo received the BS and MS degrees in computer science and information engineering and the PhD degree in computer science from the National Chiao Tung University, Hsinchu, Taiwan, in 2001, 2003, and 2008, respectively. He is currently a senior scientist at Telcordia Technologies. His current research interests are primarily in applying machine learning techniques to indoor localization, including large-scale pattern-matching localization algorithms, location tracking algorithms, and hybrid localization systems.

Yu-Chee Tseng received the PhD degree in computer and information science from Ohio State University in January 1994. He is a professor (2000-present), the chairman (2005-present), and an associate dean (2007-present) in the Department of Computer Science, National Chiao-Tung University, Taiwan. He is currently an adjunct chair professor at Chung Yuan Christian University (2006-present). He received the Outstanding Research Award by National Science Council, ROC, in both 2001-2002 and 2003-2005, the Best Paper Award by International Conference on Parallel Processing in 2003, the Elite I.T. Award in 2004, and the Distinguished Alumnus Award by Ohio State University in 2005. His research interests include mobile computing, wireless communication, and parallel and distributed computing. He serves on the editorial boards for *Telecommunication Systems* (2005-present), the *IEEE Transactions on Vehicular Technology* (2005-present), the *IEEE Transactions on Mobile Computing* (2006-present), and the *IEEE Transactions on Parallel and Distributed Systems* (2008-present).