An Online Algorithm for Fingerprint-based Location Tracking

Duc A. Tran Ting Zhang
Computer Science Department,
University of Massachusetts, Boston
Email: {duc.tran, ting.zhang001}@umb.edu

***Abstract*—We address the problem of fingerprint-based tracking of a moving device where training fingerprints are not available all at once but arrive sporadically in a stream manner. As such the only information that may be used to locate the device is its current fingerprint and the set of fingerprints obtained earlier impromptu. The challenge is to devise a localization algorithm that can scale with the fingerprint stream as this stream can grow limitlessly with time. We propose an online algorithm framework that requires storage of and computation based on only a constant-size sliding buffer of fingerprints, not all the fingerprints in the past. Our algorithm, therefore, is fast and efficient. In terms of localization error, it offers a good approximation when compared to the expensive batch algorithm that assumes access to the entire history of fingerprints.**

I. Introduction

Location information is valuable to a myriad of applications of wireless networks. GPS is the most effective way to get location information but does not work indoors. Even outdoors where this service is available, it is not energy-efficient to have to turn it on continuously all the time. Consequently, numerous efforts have been made towards GPS-free localization solutions, most adopting the fingerprint approach.

Location fingerprinting usually consists of two phases: training (offline) and positioning (online). In the offline phase, a number of sample locations are surveyed to build a map corresponding each location to a "fingerprint" which is a vector of measurements observed between the mobile device at this location and a set of "reference points" (RPs). For example, these RPs can be Wi-Fi access points [1], FM broadcasting towers [2], or cellular towers [3], and measurements can be the received signal strength observed between them and the device. In the online phase, when we need to compute a location in real time, the device's current fingerprint is compared to the training map to find the best location match. Recently, as smartphones are often shipped with various built-in sensors, fingerprint modalities other than radio have been suggested, such as sound [4] and geomagnetic field [5]. Combining these different features where they apply can lead to a rich set of discriminative features for the fingerprint information.

The calibration in the offline phase can be tedious and labor-extensive in practice. For a large area, many locations need to be surveyed to ensure good spatial coverage and many fingerprint readings need to be collected at each sample location to ensure good temporal coverage (the signal characteristics of the environment are not time invariant). In this paper, we address the problem of location fingerprint tracking in which the training fingerprints are not given all at once, but instead become available sequentially over the time. At any point in time, the only information we may use to determine the device's location is its current fingerprint and those fingerprints that have been obtained earlier. We do not make any assumption about how often training fingerprints arrive nor their timing, but do require a tracking solution that is fast and efficient so that it can scale with the (possibly infinite) fingerprint stream.

Potential applications of this problem are plentiful. A GPS-equipped smartphone, say, when being used in an urban shopping outlet, does not need to turn GPS on continuously; instead, it can be set to switch on once in a while and our algorithm can be used to compute the smartphone location during the GPS-free gaps. This results in great energy saving. In an indoor building, we can place location labels (e.g., RFID tags) at popular locations such as info desks, which the phone can read automatically when passing nearby; this labeled location information can be used to infer location at any other place. The proposed problem is also applicable to tracking of autonomous underwater vehicles (AUVs) deployed in underwater environments. Conventionally, the location of an AUV while submerged is tracked based on a built-in inertial navigation system that has to dead reckon with GPS each time the AUV surfaces. The research in this paper can be utilized to improve the localization accuracy of the AUV between these GPS fixes. It is noted that in all of the above applications, the training data comes sequentially in real time.

There are already numerous research works on mobile location tracking, but they make additional assumptions such that those about special sensors built in the device (e.g., gyroscope, accelerometer, compass, camera) [6], [7], those about mobility-specific constraints (e.g, speed, predefined map) [8] and those that are network-specific (e.g., vehicular or wireless sensor networks) [9], [10]. In contrast, we are interested in "fingerprint-based tracking" and aim to devise a tracking framework with "universal applicability" in the sense that it can work orthogonally with any type of fingerprint space; i.e., applicable where fingerprint information can be of radio signals, acoustic, or geomagnetic, etc and can contain any other information that is location-discriminative. Also, since the fingerprint stream can go to infinity, it is extremely inefficient to store and process all the fingerprints observed in

the past due to storage and computation costs. Therefore, a constraint for our problem is that we can only use a fix-size buffer of past fingerprints, not all the fingerprints.

We are not aware of any effort towards the online fingerprint-based location tracking problem with limited buffer. To solve this problem, our approach is based on the following observations about the sequence of fingerprints obtained as the device is moving: (1) *smoothness*: trajectory of a moving device in the real world should exhibit a degree of smoothness over a continuous interval of time; (2) *sparseness*: the location matrix of a mobile device over a time window exhibits a low-rank structure, as substantiated in [11], and so we conjecture that the fingerprint matrix over time should also be sparse (because of the tight correspondence between a fingerprint and its location).

Consequently, we devise a general solution framework that exploits these properties. Specifically, we make the following contributions:

- We propose to model the fingerprint-based location tracking problem as a regularized optimization problem with a regularization term to enforce smoothness in the location estimation. We substantiate this model by showing in an evaluation study the effectiveness of the objective function in order to achieve a good localization accuracy.
- We derive an optimal solution to this problem, whose computation requires access to all the fingerprints. For a limited buffer, we propose an online algorithm as an approximate solution to the optimal solution. The proposed algorithm computes the location for each fingerprint in real time such that at any point in time we require knowledge about only a sparse representation of the previously measured fingerprints. As a result, our algorithm is fast, efficient, and scalable.
- In the current state of our work, we have not been able to derive tight theoretical bounds in terms of localization accuracy, but we provide evaluation results with two real-world experiments showing that our accuracy gets incrementally better over the time and quickly converges to the accuracy of the algorithm that assumes to know all the fingerprints in the history.

The remainder of the paper is structured as follows. §II discusses the related work. §III presents the problem formulation and solution approach. The details of the proposed solution are given in §IV. Evaluation results are reported in §V. The paper is concluded in §VI with pointers to our future work.

II. Related Work

Location fingerprinting is a widely used range-free localization approach. An early adopter of this approach is Radar [1], the world's first Wi-Fi RSS-based indoor positioning system, which demonstrates the viability of using RSS information to locate a wireless device. This system works using a radio map, a lookup table that maps building locations to the corresponding RSS fingerprints empirically observed at these locations. The reference points are the Wi-Fi access points within the user's Wi-Fi range. The radio map is searched to find the closest RSS reading and the corresponding location will be used as the estimate for the user's location. Radar represents the fingerprint approach where kNN is used to determine the location. One can also employ a model-based learning approach to relate a fingerprint to a location, for example, probabilistically using Bayesian inference [12], [13] or non-probabilistically using an Artificial Neural Network (ANN) [14] or a Support Vector Machine (SVM) [15]–[17].

The above fingerprint techniques require a rich training set for the learning to be effective. When there are only a small number of sample fingerprints for training, we can utilize non-training fingerprints (those available but without known location), also called "unlabeled" fingerprints as in the area of machine learning. These unlabeled fingerprints can serve as a supplement to the labeled fingerprints (those with known location) to obtain a better leaning model, by solving a semi-supervised learning problem. A de facto standard method for semi-supervised learning is via the framework of manifold regularization originally proposed by Belkin et al. in [18]. Applying this approach to location fingerprinting [19]–[22], the location labels of the unlabeled fingerprints can be learned based on their similarity with the labeled in a manifold structure. For example, Pan et al. [19] apply manifold regularization with a Laplacian regularization term reflecting the intrinsic manifold structure of the fingerprints; here the manifold is a weighted graph of fingerprints in which the weight of an edge connecting two fingerprints represents their similarity.

Our research is different in several ways. First, we also apply a regularization framework for learning the locations of the fingerprints at unknown locations, but our regularization term is to enforce temporal smoothness in the location estimation whereas the conventional manifold regularization approach takes into account spatial smoothness. Second, we specifically address the fingerprint-based tracking problem whereas existing fingerprint techniques are designed for non-tracking localization. Third, we want an "online" algorithm with constant-bounded space and time complexities, but we are not aware of any existing fingerprint technique with this computational efficiency. Forth, as aforementioned in §I, there is a large body of research on mobile tracking [6]–[11], but they are not about fingerprint-based tracking and they make assumptions not applicable to our problem. It is also noted that sequential estimation methods such as Particle Filter and Kalman Filter have been widely used for target tracking [23]. Their purpose is to correct a trajectory given noisily observed locations, which does not apply to our case because we observe only fingerprints but not locations and need to estimate locations from these fingerprints. However, these filters can easily be integrated into our framework as a way to further enhance the location of each fingerprint after it is estimated.

III. Problem Formulation

Consider a mobile device moving according to an unknown trajectory. Its fingerprints are obtained over time in a stream manner, $\mathbf{x}_1, \mathbf{x}_2, ...$, where the time is discretized into time steps 1, 2, ... Each fingerprint is a m-dimensional point,

$\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^m$, where m is the number of fingerprint features, e.g., RSSI from different Wi-Fi APs, readings from inertial measurement units (accelerometer, gyroscope, magnetometer), and any location-discriminative feature that is available with the device, etc. Denote by $\mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^d$ the location corresponding to $\mathbf{x}_i$, where d is the location dimensionality. For ease of presentation, let $d = 1$ and so $\mathbf{y}_i = y_i$ is a real-valued scalar; the case $d > 1$ will be discussed later.

A fingerprint is said to be "labeled" if it is accompanied with a known location and "unlabeled" otherwise. A fingerprint may be observed with or without a location label, which is unpredictable. When a labeled fingerprint is available, it will be a new training fingerprint for our algorithm. Therefore, although we do not have all the training fingerprints available at once, they come impromptu during the fingerprint stream. Notation h_i represents whether a fingerprint x_i is labeled ($h_i = 1$) or unlabeled ($h_i = 0$). We want to determine a real-time location estimator $f : \mathcal{X} \rightarrow \mathbb{R}$ so that the estimated location at time t for the sequence of fingerprints $[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t]$ will be $f(\mathbf{x}_t)$ which, ideally, should equal the ground-truth y_t (unknown except for the labeled fingerprints).

A. Assumptions and Constraints

Our assumption regarding the availability of labeled fingerprints during the fingerprint stream is reasonable because (1) it is necessary: localization is impossible otherwise, and (2) it is practically doable: these location labels can be made available, for example, by crowdsourcing over the time or placing a number of "device-readable" location labels (e.g., RFID tags) in different locations in the area so a mobile device can read when it is traveling nearby. In application scenarios mentioned in §I, the labeled fingerprints are obtained when the AUV surfaces to get GPS fixes or when the GPS is enabled on a smartphone (GPS previously turned off due to energy saving or lost in harsh environments).

The main constraint we set for the online fingerprint tracking problem is that at any positioning time we can afford only a constant-size budget of fingerprint information, based on which location computation is performed. In other words, at any time t, the only new information we receive is the new fingerprint $\mathbf{x}_t$ and all the information we can use for the computation is just a constant-size representation of the fingerprint history including the new fingerprint. As such at no time can we assume to know about all the fingerprints observed in the past.

B. Optimization Approach

We formulate the problem as a regularization problem assuming to know all the fingerprints and then seek an online solution with constant-bounded space and time complexities to best approximate the exact solution to this problem.

Ideally, the estimated location of a labeled fingerprint should match its given location. Therefore, our first goal is to minimize the estimation error with respect to the labeled fingerprints. This is quantified by minimizing

$$\min_f \left\{ E(f) = \sum_{i=1}^{t} h_i (f(\mathbf{x}_i) - y_i)^2 \right\}. \quad (1)$$

Our second goal is to maximize the temporal smoothness in the fingerprint space. The fingerprints obtained in consecutive moves of the device should correspond to nearby locations whose distance should be constant between accelerations. To exploit this property, we want the location estimator to minimize the following quadratic functional:

$$\min_f \left\{ T(f) = \sum_{i=3}^{t} (f(\mathbf{x}_i) + f(\mathbf{x}_{i-2}) - 2f(\mathbf{x}_{i-1}))^2 \right\}. \quad (2)$$

In practice, other optimization goals may be introduced to better satisfy the kinetic properties of the deployed application. Since we want to devise a general framework, without such knowledge, we focus only on the two goals above. To optimize these individual goals, we combine them into a single risk function. Specifically, the location estimator f we are seeking is a function that belongs to $\mathcal{H}_K$, the reproducing kernel Hilbert space (RKHS) associated with a positive definite kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{|\mathbf{x}_i - \mathbf{x}_j|^2}{2\gamma^2}\right)$ (for some constant γ), implementing the inner product, and that is the solution minimizing the following regularized empirical risk:

$$\min_f \left\{ J(f) = E(f) + \lambda_T T(f) + \lambda_K \|f\|_K^2 \right\}, \quad (3)$$

where the last term is added to make the minimization problem well-posed; here, f is preferred to be smooth with respect to kernel K (the norm in this term is defined using the inner product associated with the kernel) and coefficient $\lambda_K > 0$ can be set to a small value, sufficiently small to make the optimization solvable. The regularization coefficient $\lambda_T \in (0, \infty)$ regulates the importance of temporal smoothness in the optimization of the risk.

It is noted that our problem is similar to the semi-supervised learning problem in machine learning insofar as we need to predict labels for unlabeled points given a set of points, labeled and unlabeled. Therefore, an intuitive approach is to use manifold regularization [18] as in earlier work on fingerprint-based (non-tracking) localization [19], [20]. Our evaluation study will later show this framework less effective than our regularization framework.

C. Optimality

Theoretically, an optimal solution can be found for f that minimizes the $J(f)$ in Eq. (3) at any time t if we know all the fingerprints observed from time 1 to time t. First, we denote the following matrices: $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_t)]^\intercal$, $\mathbf{y} = [y_1, y_2, ..., y_t]^\intercal$ (y_i is set to zero by default for unlabeled $\mathbf{x}_i$), $\mathbf{H} = diag(h_1, h_2, ..., h_t)$, the identity matrix $\mathbf{I} = diag(\underbrace{1, 1, ..., 1}_{t})$, and $\mathbf{B} = [b_{ij}]_{t \times t} = \mathbf{D}^\intercal \mathbf{D}$ where $\mathbf{D}$

is the second-order difference matrix

$$\mathbf{D} = \left[d_{ij} = \begin{cases} 1 & \text{if } (i \geq 3) \wedge (j = i \vee j = i-2) \\ -2 & \text{if } (i \geq 3) \wedge (j = i-1) \\ 0 & \text{otherwise} \end{cases} \right]_{t \times t}$$

Then, we can compute the optimal $\mathbf{f}$ as follows.

Proposition III.1. *The minimizer of risk $J(f)$ admits the following solution, in matrix form,*

$$\mathbf{f} = (\lambda_K \mathbf{I} + \mathbf{KH} + \lambda_T \mathbf{KB})^{-1} \mathbf{KHy}. \tag{4}$$

Proof: Express the functionals in Eqs. (1) and (2) in matrix form as follows: $E(f) = (\mathbf{f}-\mathbf{y})^\intercal \mathbf{H}(\mathbf{f}-\mathbf{y})$, $T(f) = \mathbf{f}^\intercal \mathbf{B} \mathbf{f}$. In the matrix form, $\|\mathbf{f}\|_K^2 = \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha}$. Thus, the risk $J(f)$ in Eq. (3) can be expressed in matrix form,

$$\begin{aligned} J(f) &= (\mathbf{f}-\mathbf{y})^\intercal \mathbf{H}(\mathbf{f}-\mathbf{y}) + \lambda_T \mathbf{f}^\intercal \mathbf{B} \mathbf{f} + \lambda_K \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha} \\ &= \mathbf{f}^\intercal (\mathbf{H} + \lambda_T \mathbf{B}) \mathbf{f} - 2\mathbf{y}^\intercal \mathbf{H} \mathbf{f} + \mathbf{y}^\intercal \mathbf{H} \mathbf{y} + \lambda_K \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha} \\ &= \boldsymbol{\alpha}^\intercal \mathbf{Q} \boldsymbol{\alpha} - 2\mathbf{y}^\intercal \mathbf{H} \mathbf{K}^\intercal \boldsymbol{\alpha} + \mathbf{y}^\intercal \mathbf{H} \mathbf{y}. \end{aligned}$$

where $\mathbf{Q} = \mathbf{K}\left(\lambda_K \mathbf{I} + (\mathbf{H} + \lambda_T \mathbf{B}) \mathbf{K}^\intercal\right)$. To minimize J, set its derivative with respect to $\boldsymbol{\alpha}$ to zero,

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = (\mathbf{Q} + \mathbf{Q}^\intercal)\boldsymbol{\alpha} - 2\mathbf{KHy} = 0. \tag{5}$$

Since $\mathbf{K}$, $\mathbf{H}$, $\mathbf{B}$, and $\mathbf{L}$ are symmetric matrices, we have $\mathbf{Q}^\intercal = \mathbf{Q}$ and so Eq. (5) becomes

$$\begin{aligned} 2\left(\lambda_K \mathbf{I} + \mathbf{K}(\mathbf{H} + \lambda_T \mathbf{B})\right) \mathbf{K} \boldsymbol{\alpha} - 2\mathbf{KHy} &= 0 \\ \left(\lambda_K \mathbf{I} + \mathbf{K}(\mathbf{H} + \lambda_T \mathbf{B})\right) \mathbf{K} \boldsymbol{\alpha} - \mathbf{KHy} &= 0 \\ \Rightarrow \mathbf{f} = (\lambda_K \mathbf{I} + \mathbf{KH} + \lambda_T \mathbf{KB})^{-1} \mathbf{KHy}. & \end{aligned}$$

■

The computation of Eq. (4) runs in $O(t^3)$ time because it involves $(t \times t)$-matrix inverse. This computation requires access to all the fingerprints in the history. Furthermore, it has to be re-computed each time t is increased to $t+1$, $t+2$, etc. In the next section, we propose an approximate algorithm using only a fixed budget of information, which can scale as the fingerprint stream grows ad infinitum.

IV. Online Algorithm

As aforementioned in §I, the fingerprint space should be sparse in both time and space. This suggests that it may be possible to approximate the growing set of fingerprints with only a sparse representation which we can use to estimate location in real time. Below we present our idea first, followed by further algorithmic details.

Let us represent the set of fingerprints $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t$ compactly as a multi-set $\{(\bar{\mathbf{x}}_1, m_1), (\bar{\mathbf{x}}_2, m_2), ..., (\bar{\mathbf{x}}_k, m_k)\}$ where the representative elements are $\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, ..., \bar{\mathbf{x}}_k$ and their respective multiplicities $m_1, m_2, ..., m_k$ ($\sum_{i=1}^k m_i = t$). In other words, there are m_1 fingerprints with value $\bar{\mathbf{x}}_1$, m_2 fingerprints with value $\bar{\mathbf{x}}_2$, etc. Let $\bar{\mathbf{f}} = [f(\bar{\mathbf{x}}_1), f(\bar{\mathbf{x}}_2), ..., f(\bar{\mathbf{x}}_k)]^\intercal$ and $\bar{\mathbf{y}} = [\bar{y}_1, \bar{y}_2, ..., \bar{y}_k]^\intercal$ be the corresponding compact vectors for estimated locations and ground-truth locations, respectively. Our approach is to transform the original optimization problem to a more compact version using only the knowledge about the representative fingerprints.

We rewrite the estimation error with respect to the labeled fingerprints $E(f)$ as

$$\begin{aligned} E(f) &= \sum_{i=1}^{t} h_i (f(\mathbf{x}_i) - y_i)^2 \\ &= \sum_{i=1}^{k} \bar{h}_i (f(\bar{\mathbf{x}}_i) - \bar{y}_i)^2 = (\bar{\mathbf{f}} - \bar{\mathbf{y}})^\intercal \bar{\mathbf{H}} (\bar{\mathbf{f}} - \bar{\mathbf{y}}) \end{aligned}$$

where $\bar{\mathbf{H}} = diag(\bar{h}_1, \bar{h}_2, ..., \bar{h}_k)$ and $\bar{h}_i = m_i$ if fingerprint $\bar{\mathbf{x}}_i$ is labeled with corresponding known location $\bar{y}_i$ and $\bar{h}_i = 0$ otherwise.

We rewrite the regularization term $T(f)$ as

$$\begin{aligned} T(f) &= \mathbf{f}^\intercal \mathbf{B} \mathbf{f} = \sum_{i=1}^{t} f(\mathbf{x}_i) \sum_{j=1}^{t} f(\mathbf{x}_j) b_{ij} \\ &= \sum_{i=1}^{t} f(\mathbf{x}_i) \sum_{j=1}^{k} f(\bar{\mathbf{x}}_j) \sum_{q | \mathbf{x_q} = \bar{\mathbf{x}}_j} b_{iq} \\ &= \sum_{j=1}^{k} f(\bar{\mathbf{x}}_j) \sum_{i=1}^{t} f(\mathbf{x}_i) \sum_{q | \mathbf{x_q} = \bar{\mathbf{x}}_j} b_{iq} \\ &= \sum_{j=1}^{k} f(\bar{\mathbf{x}}_j) \sum_{i=1}^{k} f(\bar{\mathbf{x}}_i) \left(\underbrace{\sum_{p | \mathbf{x_p} = \bar{\mathbf{x}}_i} \sum_{q | \mathbf{x_q} = \bar{\mathbf{x}}_j} b_{pq}}_{\bar{b}_{ij}} \right) \\ &= \sum_{i=1}^{k} \sum_{j=k}^{k} f(\bar{\mathbf{x}}_i) \bar{b}_{ij} f(\bar{\mathbf{x}}_j) = \bar{\mathbf{f}}^\intercal \bar{\mathbf{B}} \bar{\mathbf{f}} \end{aligned}$$

where $\bar{\mathbf{B}} = [\bar{b}_{ij}]_{k \times k}$. Intuitively, $\bar{\mathbf{B}}$ is a compact version of $\mathbf{B}$ by merging (summing up values of) all the entries in $\mathbf{B}$ whose row/column indices correspond to fingerprints having identical values.

Consequently, the original minimization problem (3) is equivalent to an exact same minimization problem except that the given fingerprints are $\{\bar{\mathbf{x}}_i\}$ instead of $\{\mathbf{x}_i\}$ and the $k \times k$ matrices $\bar{\mathbf{H}}$, $\bar{\mathbf{B}}$, and $\bar{\mathbf{K}} = \left[\bar{k}_{ij} = K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_i)\right]_{k \times k}$ are used instead of $\mathbf{H}$, $\mathbf{B}$, and $\mathbf{K}$, respectively. Applying Proposition III.1 on the latter problem, which is more compact, we obtain the following result.

Corollary IV.1. *The minimizer of risk $J(f)$ admits the solution, in matrix form, $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_t)]$ where its i^{th} entry is the j^{th} entry of the column vector*

$$\bar{\mathbf{f}} = \left(\lambda_K \bar{\mathbf{I}} + \bar{\mathbf{K}} \bar{\mathbf{H}} + \lambda_T \bar{\mathbf{K}} \bar{\mathbf{B}}\right)^{-1} \bar{\mathbf{K}} \bar{\mathbf{H}} \bar{\mathbf{y}}, \tag{6}$$

such that $\mathbf{x}_i = \bar{\mathbf{x}}_j$. (Here, $\bar{\mathbf{I}}$ is the identity matrix of size $k \times k$.)

This corollary suggests that instead of using a $O(t^3)$-time and $O(t^2)$-space algorithm for computing Eq. (4) we can

obtain an exact solution by computing Eq. (6), which involves manipulation of matrices of size $k \times k$ only; hence, $O(k^3)$-time and $O(k^2)$-space complexities. Therefore, if k is kept small, we have a much more efficient algorithm. However, if we use the exact multi-set representation for the set of fingerprints, we cannot predict the growth of k as t increases. Therefore, we approximate this set with a fix-sized buffer of up to k (constant) representative fingerprints and perform location estimation based on the content of this buffer only. Algorithmic details are provided below.

A. Representative Buffer

At any time t we maintain a buffer $B_t = \{(\tilde{\mathbf{x}}_i, \tilde{m}_i, \tilde{y}_i)\}_i$ of up to k (constant) representative fingerprints $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_k \in \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\}$, where there is a count $\tilde{m}_i$ for $\tilde{\mathbf{x}}_i$, counting the number of fingerprints it represents. In this approximation, each fingerprint is approximated by its closest representative. Here, $\tilde{y}_i$ is the location of $\tilde{\mathbf{x}}_i$ if it is labeled and zero otherwise. We use the notation ($\tilde{}$) to mean "approximate" to distinguish from ($\bar{}$) which means "exact".

The representative fingerprints in the buffer are the k centers, as in the well-known metric k-center clustering problem, of the fingerprint set, i.e.,

$$[\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_k] = \underset{[\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_k]}{\operatorname{argmin}} \left(\max_{1 \le j \le t} \left(\min_{1 \le i \le k} \|\mathbf{x}_j - \tilde{\mathbf{x}}_i\| \right) \right).$$

We use the k-center representatives because this is a good compact representation of the fingerprint graph in terms of spatial representativity (unlikely for two fingerprints with similar values to be both selected centers) and temporal representativity (unlikely for two fingerprints observed in similar times to be both selected centers).

To update these k centers incrementally each time a new fingerprint is added, we derive an algorithm based on a modified version of a well-known incremental k-center algorithm, the Doubling Algorithm by Charikar et al. [24], with one revision that we prefer labeled fingerprints to serve as centers. Specifically, we enforce two rules: (1) when the new fingerprint needs to join an existing cluster, it prefers to be represented by a center that is labeled, and (2) in the re-clustering process when the number of centers exceeds k, labeled fingerprints are selected first to serve as center. More presence of labeled fingerprints in the representative set should result in more information useful for the location prediction.

B. Location Estimation

Similar to how we denote the matrices $\bar{\mathbf{H}}$ and $\bar{\mathbf{B}}$ for the exact multi-set, we denote the corresponding matrices $\tilde{\mathbf{H}}$ and $\tilde{\mathbf{B}}$ for the set of fingerprints in B_t. Matrix $\tilde{\mathbf{H}}$ simply is $diag(\tilde{h}_1, \tilde{h}_2, ..., \tilde{h}_k)$ where $\tilde{h}_i = \tilde{m}_i$ if fingerprint $\tilde{\mathbf{x}}_i$ is labeled and $\tilde{h}_i = 0$ otherwise. To compute the matrix $\tilde{\mathbf{B}}$, we update its entries $\{\tilde{b}_{ij}\}_{k \times k}$ incrementally during the same time as the buffer is being updated. Initially, at time zero, $\tilde{b}_{ij}$ is set to 0 for every $i, j \le k$. Suppose that the current buffer is $\{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_k\}$ when we receive a new fingerprint $\mathbf{x}_t$. As discussed earlier, the buffer update consists in two steps: (1) assign a representative to the new fingerprint and (2) re-cluster the representatives if necessary (when there are more than k of them).

In the first step, suppose that $\mathbf{x}_t$ is assigned to some representative $\tilde{\mathbf{x}}_i$. Because of this new assignment, only $\{\tilde{b}_{ij}, \tilde{b}_{ji}\}$ for all $j \le k$ need to be updated. This update simply is

$$\tilde{b}_{ij} \mathrel{+}= \sum_{q | \mathbf{x_q} \mapsto \tilde{\mathbf{x}}_j} b_{tq}, \; \tilde{b}_{ji} \mathrel{+}= \sum_{q | \mathbf{x_q} \mapsto \tilde{\mathbf{x}}_j} b_{tq}.$$

Here, "$\mapsto$" denotes the assignment of a fingerprint to a representative. Note that $\mathbf{B} = \mathbf{D}\mathbf{D}^\intercal$ is the following pentadiagonal matrix

$$\begin{bmatrix} 1 & -2 & 1 & 0 & ... & ... & ... & ... & 0 \\ -2 & 5 & -4 & 1 & 0 & ... & ... & ... & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & ... & ... & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 & ... & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & ... & ... & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & ... & ... & ... & 0 & 1 & -4 & 5 & -2 \\ 0 & ... & ... & ... & ... & 0 & 1 & -2 & 1 \end{bmatrix}_{t \times t}$$

where all its entries b_{pq} is zero except when $|p - q| \le 2$. Therefore, $\tilde{b}_{ij}$ (and $\tilde{b}_{ji}$) can be efficiently computed as

$$\tilde{b}_{ij} \mathrel{+}= \begin{cases} b_{t-1,t} & \text{if } \mathbf{x}_{t-1} \mapsto \tilde{\mathbf{x}}_j \wedge \mathbf{x}_{t-2} \not\mapsto \tilde{\mathbf{x}}_j \\ b_{t-2,t} & \text{if } \mathbf{x}_{t-1} \not\mapsto \tilde{\mathbf{x}}_j \wedge \mathbf{x}_{t-2} \mapsto \tilde{\mathbf{x}}_j \\ b_{t-1,t} + b_{t-2,t} & \text{if } \mathbf{x}_{t-1} \mapsto \tilde{\mathbf{x}}_j \wedge \mathbf{x}_{t-2} \mapsto \tilde{\mathbf{x}}_j \\ 0 & \text{otherwise.} \end{cases}$$

For this computation to be possible, in addition to the buffer itself, we also always keep track of which fingerprints in the buffer represent the two latest fingerprints ($\mathbf{x}_{t-1}$, $\mathbf{x}_{t-2}$). It is also important to note that at each new step t, the entry $b_{t-1,t-1}$ changes value from 1 (earlier value at time $t-1$) to 5 (value at time t). Similarly, $b_{t-2,t-2}$ changes value from 5 to 6, and $b_{t-1,t-2}$ and $b_{t-2,t-1}$ both from -2 to -4. The values of the other entries of matrix B are not changed. Consequently, supposing that $\mathbf{x}_{t-1} \mapsto \tilde{\mathbf{x}}_{j_1}$ and $\mathbf{x}_{t-2} \mapsto \tilde{\mathbf{x}}_{j_2}$, we need to make the following updates:

$$\tilde{b}_{j_1 j_1} \mathrel{+}= 4, \; \tilde{b}_{j_2 j_2} \mathrel{+}= 1, \tilde{b}_{j_1 j_2} \mathrel{-}= 2, \; \tilde{b}_{j_2 j_1} \mathrel{-}= 2.$$

The total time for updating $\tilde{\mathbf{B}}$ in this step is $O(k)$.

In the second step, which takes place only if there are more than k representative fingerprints in the buffer, we need to re-cluster them. This step involves merging of the representatives. Each time a representative $\tilde{\mathbf{x}}_l$ is collapsed into another representative $\tilde{\mathbf{x}}_i$ (lines 25-26), we make the following update:

$$\begin{aligned} \forall\, 1 \le j \le k,\; j \ne l \quad &: \quad \tilde{b}_{ij} \mathrel{+}= \tilde{b}_{jl}, \;\; \tilde{b}_{ji} \mathrel{+}= \tilde{b}_{jl} \\ \forall\, 1 \le j \le k \quad &: \quad \tilde{b}_{lj} = \tilde{b}_{jl} = 0 \end{aligned}$$

The time complexity for updating $\tilde{\mathbf{B}}$ in the re-clustering step, if it occurs, is also $O(k)$.

After matrix $\tilde{\mathbf{B}}$ has been updated the location estimation is simply an application of Eq. (6) in Corollary IV.1 where

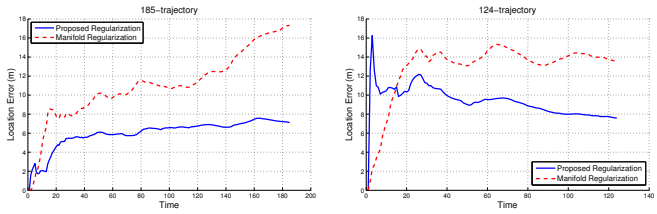
we use $\tilde{\mathbf{H}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{K}}$ instead of $\bar{\mathbf{H}}$, $\bar{\mathbf{B}}$, and $\bar{\mathbf{K}}$. The total time complexity for estimate the location of each fingerprint is dominated by the time to compute inverse matrices; hence, $O(k^3)$. We have so far assumed that the location is 1D. For 2D or 3D localization, we simply apply the same algorithm separately for each coordinate.

Our algorithm is a parametric algorithm as it assumes given values of two main parameters, (1) k: the number of k-centers as a sparse representation of the whole fingerprint set; and (2) λ_T: the regularization coefficient for temporal smoothness. The localization error depends on how we choose values for these parameters. The choice of k provides a tradeoff between efficiency and accuracy; a larger value is better for accuracy but worse for computation. Therefore, this decision should be made by the application. For example, one could consider a pay-for-quality policy: additional fee is charged for better resolution of tracking accuracy. Choosing λ_T in an automated way is not trivial, which is the same challenge as often seen in regularization frameworks. In any case, λ_T should depend on the label rate reciprocally; the more often we see a labeled fingerprint in the stream, the less smoothing is needed.

V. Evaluation

Our evaluation study is focused on validation of our regularization framework and with respect to the optimization goal in this framework how well the proposed online algorithm approximates the batch algorithm. The batch algorithm assumes an infinite buffer and so at the time of estimating the location for each fingerprint this algorithm has access to all the fingerprints previously obtained. The localization time and error are the metrics for comparison. The localization time is computed as the average time to compute the location for each individual unlabeled fingerprint. The location error (up to time t) is computed as the average "individual" location error for unlabeled fingerprints over the path traveled from the beginning (up to time t). The "individual" error corresponding to a fingerprint $\mathbf{x}_t$ is the Euclidean distance between its location estimate at time t and its ground-truth location.

The evaluation was conducted with two different trajectories, shown in Figure 1, experimented on the floor of the Computer Science (CS) department and on the upper-level floor of the Campus Center (CC) at our university. The CS-trajectory (Figure 1(a)) is a sequence of 185 Wi-Fi RSSI fingerprints at 185 locations (24 APs heard per location). The CC-trajectory (Figure 1(b)) is a sequence of 124 Wi-Fi RSSI fingerprints (39 APs heard per location). In each trajectory, the labeling status of each point is determined based on a label rate $p_l \in \{0.1$ (low), 0.5 (medium), 0.9 (high)$\}$. For each choice of p_l, the results are averaged over five random runs. The range of values for the regularization coefficient is $\lambda_T \in \{0, 10^{-8}, 10^{-7}, ..., 10^{-1}, 1\}$, representing ten different scales. Parameter k varies from 10% to 100% of the sequence length. Parameter γ in the kernel function is set to 1 and parameter λ_K is set to 10^{-6} which is small enough to make our minimization problem solvable.



(a) CS-Trajectory (b) CC-Trajectory

Fig. 2. Comparison to manifold regularization in terms of location error over time, averaged across all cases of label rate.

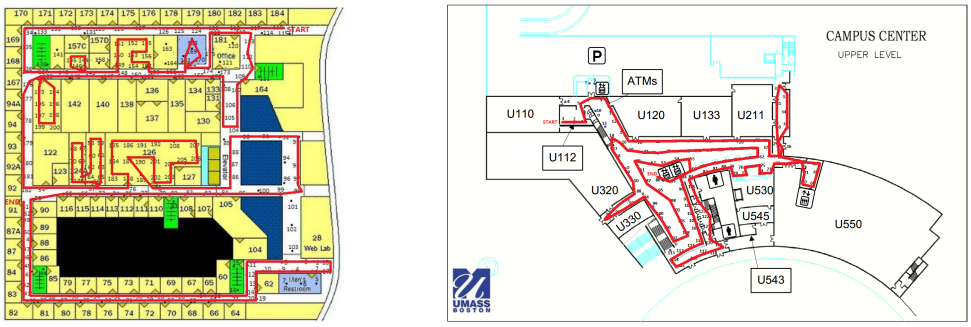
A. Validation of the Regularization Framework

As discussed in §III-B, manifold regularization (MR) [18] is used in earlier work on fingerprint-based localization [19], [20]. The manifold approach is aimed to minimize the objective function $\min_f \{J(f) = E(f) + \lambda_S S(f) + \lambda_K \|f\|_K^2\}$, where the regularization term $S(f)$ represents the property that fingerprints of similar values should be located nearby, $S(f) = \sum_{i=1}^{t}\sum_{j=1}^{i} w(\mathbf{x}_i, \mathbf{x}_j)(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2$; here, $w(\mathbf{x}_i, \mathbf{x}_j)$ is a similarity measure between $\mathbf{x}_i$ and $\mathbf{x}_j$ and the coefficient $\lambda_S \in (0, \infty)$ controls the importance of minimizing $S(f)$. Our regularization approach uses a different regularization term, $T(f)$ in Eq. (2), to regularize temporal smoothness.

We compare these two regularization frameworks in terms of how effective they are leading to good localization accuracy, using their respectively best coefficients λ_T (our framework) and λ_S (MR), found as a result of an offline cross-validation procedure. Figure 2 provides a summary of the comparison for both the CS-trajectory and CC-trajectory, showing the location error over the time, which is averaged across all cases of label rate. It is observed that throughout the travel path of the device, our regularization approach results in substantially better location error. Another observation favoring our framework is that its error converges to a stable value quickly as more fingerprints are observed whereas MR's error keeps increasing before showing any sign of convergence. These results validate the use of the temporal smoothing term in our regularization framework. Next, we turn the focus to the comparison between the proposed algorithm and the batch algorithm.

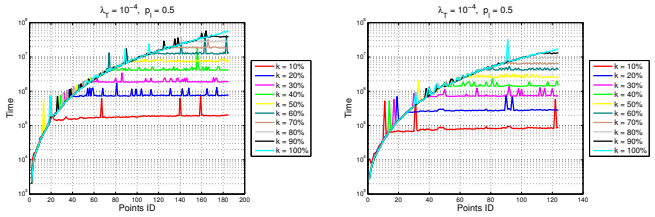
B. Localization Time is an Obvious Advantage

The main advantage of the online algorithm is its computation time. It has $O(k^3)$ time complexity, which is constant with respect to the time t, whereas the batch algorithm's time complexity is $O(t^3)$. Figure 3 provides an illustration of this advantage for both trajectories in the study; here the time information is shown in logarithmic scale for $\lambda_T = 10^{-4}$ and $p_l = 0.5$ but similar results are observed for the other choices of these parameters. As expected, the localization time exhibits a cubic growth until the buffer reaches its capacity (k in the online algorithm and no limit in the batch algorithm). Regardless of how large the buffer is, as long as its size is fixed, eventually the computation time will converge to a stable value that does not increase even when the fingerprint



(a) CS-Trajectory (185 locations) (b) CC-Trajectory (124 locations)

Fig. 1. Two trajectories are shown in red color, obtained in two different areas: (a) on the floor of the CS department (68m × 63m); (b) on the upper-level floor in the campus center (box dimension 185m × 113m) at UMass Boston.



(a) CS-Trajectory (b) CC-Trajectory

Fig. 3. Localization time (in log scale) to locate each individual fingerprint.

stream gets longer. The cubic growth in time renders the batch algorithm extremely inefficient in practice where the tracking is required for an extended period of time.

C. Location Accuracy Can Compete

While the computation time advantage is obvious, the trade-off, as in any online algorithm, is possible loss in accuracy. In our study, we have found that in spite of using less information to make prediction, our algorithm can be as accurate as the batch algorithm. For example, Figure 4 shows that for CS-Trajectory when 50% of the fingerprint stream is unlabeled, using a buffer only 20% of the fingerprint stream length, the online algorithm results in a trajectory (Figure 4(a)) almost identical to that of the batch algorithm (Figure 4(b)), and this trajectory closely resembles the ground-truth trajectory (Figure 4(c)); similar result is obtained for CC-Trajectory. Location estimates for the first few fingerprints are not good due to lack of labeled information but the accuracy improves over the time. Although these results represent only one case of study where λ_T is set to 10^{-4}, it is an evidence that it is possible to achieve comparable tracking quality even when using only a constant-size buffer of the fingerprint stream. It is noted that here we plot the location estimated for each sequential fingerprint instantly at the time it is observed. If we want, we can use the latest location estimator at any point in time to obtain even better estimates for all the unlabeled fingerprints in the past up to this time. These enhanced estimates are useful if there is a need for a posterior fix of the trajectory.

D. Analysis of Location Error

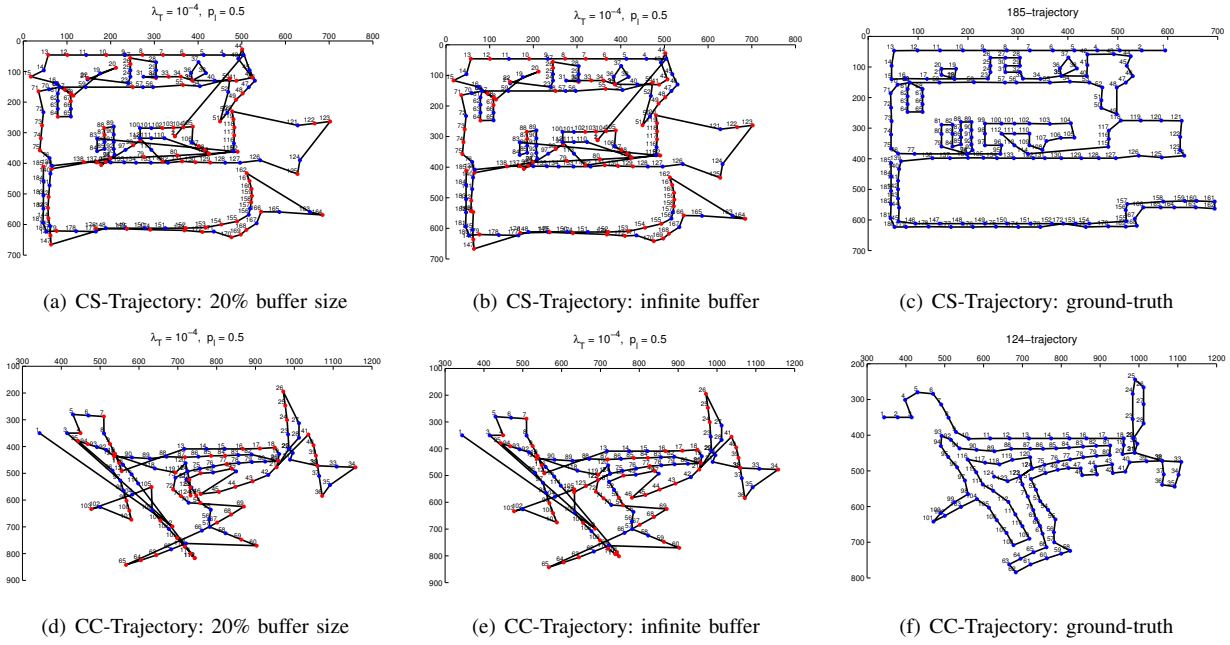
Numerical analysis of location error with different parameter choices is discussed in this section. Figure 5 shows the location error for both CS-Trajectory and CC-Trajectory as the buffer size changes, for difference choices of λ_T and p_l. It is expected that the error should decrease as the label rate increases, but we want to emphasize the importance of choosing an appropriate value for λ_T. It is observed that temporal smoothness regularization ($\lambda_T > 0$ vs. $\lambda_T = 0$) does help with the localization accuracy. On the other hand, too much regularization ($\lambda = 1$) does not offer the best result. The best result is obtained when the right extent of regularization is enforced. The best choice seems to be $\lambda_T \in (10^{-5}, 10^{-4})$ for either trajectory. With this choice, depending on the label rate (low/medium/high), we can obtain an error as good as 15m/5m/3.5m for CS-Trajectory (Figures 5(a, b, c)) and 20m/10m/3m for CC-Trajectory (Figures 5(d, e, f)).

Comparing the online algorithm with the batch algorithm, Figures 5(g, h, i) show the ratio between the location error of the online algorithm and that of the batch algorithm for all choices of λ_T for both CS-Trajectory and CC-Trajectory. What is noticeable here is that the online algorithm can be well-approximative of the batch algorithm. The approximation factor quickly approaches 1 when the buffer size increases ($k \geq 30\%$). Especially, when the label rate is high enough, e.g., when $p_l = 0.5$, this factor is less than 1.2 for CS-Trajectory and less than 1.6 for CC-Trajectory, even when the buffer size is only 10% of the length of the fingerprint stream.

It is noted that while our way to achieve good tracking accuracy is by minimizing the regularized risk, this approach is just a heuristic. It is not an absolute guarantee that if this risk is minimized than the localization accuracy will be optimal. This explains why in some cases of CS-Trajectory, as seen in Figures 5(b, c), the online algorithm actually outperforms the batch algorithm, which makes the former even more favorable.

VI. Conclusions

We have addressed the problem of real-time fingerprint-based tracking of mobile devices, assuming that the training data arrives in a stream manner instead of being available



(a) CS-Trajectory: 20% buffer size (b) CS-Trajectory: infinite buffer (c) CS-Trajectory: ground-truth

(d) CC-Trajectory: 20% buffer size (e) CC-Trajectory: infinite buffer (f) CC-Trajectory: ground-truth

Fig. 4. Drawing of the estimated trajectory: Red-colored points are location estimates for unlabeled fingerprints and blue-colored points are the ground-truth locations of the labeled fingerprints. The numbers represent the ID of the fingerprints sorted in time of observation.

all at once and requiring that only a limited buffer may be used for processing. We have proposed an online algorithm framework based on two key ideas: formulate the tracking problem as an optimization problem with a regularization term to regulate temporal smoothness, a fingerprint property evident in real-world mobility, and solve this optimization problem approximately using only a constant-bounded buffer of information which is a sparse representation of the entire fingerprint stream. A clear advantage is in the computational efficiency; our algorithm can scale regardlessly of however long the fingerprint stream can grow. In terms of localization error, our evaluation study has shown that the proposed framework can offer a convincing approximation to the batch algorithm approach. As the latter has limited use in practice due to its prohibitive computational costs, our research offers an alternative solution that is substantially faster and more efficient. A key challenge remains though; that is, we need a way to automatically determine the regularization coefficient. This will be the focus of our immediate future work.

ACKNOWLEDGEMENTS

This work was supported by the NSF award CNS-1116430.

REFERENCES

[1] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *INFOCOM*, 2000, pp. 775–784.

[2] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha, "Fm-based indoor localization," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 169–182. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307653

[3] A. Varshavsky, E. de Lara, J. Hightower, A. LaMarca, and V. Otsason, "Gsm indoor localization," *Pervasive Mob. Comput.*, vol. 3, no. 6, pp. 698–720, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.pmcj.2007.07.004

[4] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik, "Indoor localization without infrastructure using the acoustic background spectrum," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 155–168. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000011

[5] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 141–154. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000010

[6] C. Wu, Z. Yang, Y. Liu, and W. Xi, "Will: Wireless indoor localization without site survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 4, pp. 839–848, 2013.

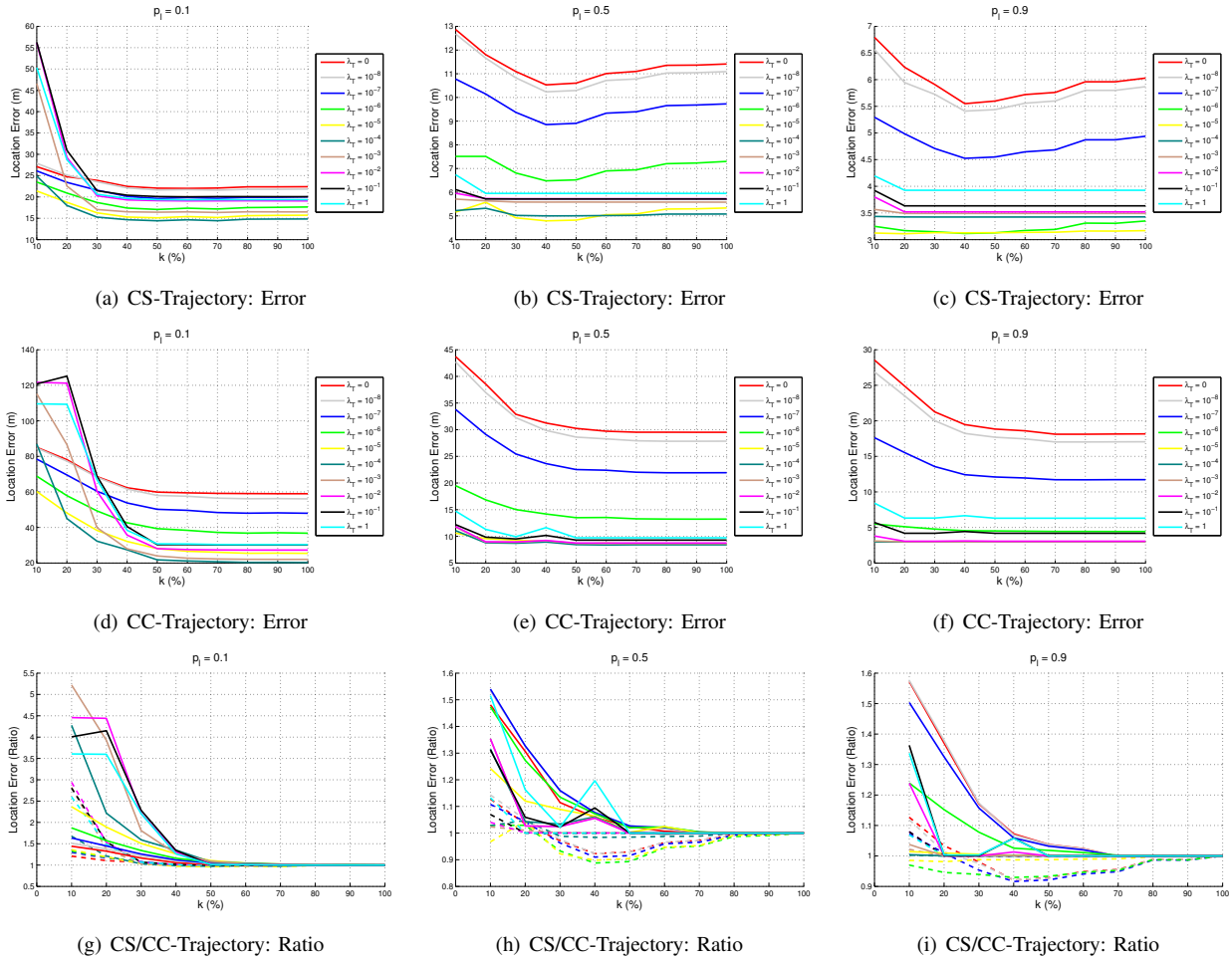
[7] O. Woodman and R. Harle, "Pedestrian localisation for indoor environments," in *Proceedings of the 10th International Conference on Ubiquitous Computing*, ser. UbiComp '08. New York, NY, USA: ACM, 2008, pp. 114–123. [Online]. Available: http://doi.acm.org/10.1145/1409635.1409651

[8] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '04. New York, NY, USA: ACM, 2004, pp. 45–57. [Online]. Available: http://doi.acm.org/10.1145/1023720.1023726

[9] A. Boukerche, H. A. B. F. Oliveira, E. F. Nakamura, and A. A. F. Loureiro, "Vehicular ad hoc networks: A new challenge for localization-based systems," *Comput. Commun.*, vol. 31, no. 12, pp. 2838–2849, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2007.12.004

[10] I. Amundson and X. D. Koutsoukos, "A survey on localization for mobile wireless sensor networks," in *Proceedings of the 2Nd International Conference on Mobile Entity Localization and Tracking in GPS-less Environments*, ser. MELT'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 235–254. [Online]. Available: http://dl.acm.org/citation.cfm?id=1813141.1813162

[11] S. Rallapalli, L. Qiu, Y. Zhang, and Y.-C. Chen, "Exploiting temporal stability and low-rank structure for localization in mobile networks," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, ser. MobiCom '10. New York, NY, USA: ACM, 2010, pp. 161–172. [Online]. Available:



(a) CS-Trajectory: Error (b) CS-Trajectory: Error (c) CS-Trajectory: Error

(d) CC-Trajectory: Error (e) CC-Trajectory: Error (f) CC-Trajectory: Error

(g) CS/CC-Trajectory: Ratio (h) CS/CC-Trajectory: Ratio (i) CS/CC-Trajectory: Ratio

Fig. 5. Location error for CS-Trajectory (a, b, c) and CC-Trajectory (d, e, f) as buffer size increases; (g, h, i) Ratio of location error relative to the batch algorithm's error as buffer size increases, where each curve corresponds to CS-Trajectory (dashed-line) or CC-Trajectory (solid-line) in each case of λ_T.

http://doi.acm.org/10.1145/1859995.1860015

[12] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, "A probabilistic approach to WLAN user location estimation," *International Journal of Wireless Information Networks*, vol. 9, no. 3, pp. 155–164, July 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1016003126882

[13] M. Youssef, A. Agrawala, and A. U. Shankar, "Wlan location determination via clustering and probability distributions," in *In IEEE PerCom 2003*, 2003.

[14] C. Laoudias, D. G. Eliades, P. Kemppi, C. G. Panayiotou, and M. M. Polycarpou, "Indoor localization using neural networks with location fingerprints," in *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II*, ser. ICANN '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 954–963.

[15] M. Brunato and R. Battiti, "Statistical learning theory for location fingerprinting in wireless lans," *Comput. Netw.*, vol. 47, no. 6, pp. 825–845, Apr. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2004.09.004

[16] C.-L. Wu, L.-C. Fu, and F.-L. Lian, "WLAN location determination in e-home via support vector classification," in *Networking, Sensing and Control, 2004 IEEE International Conference on*, vol. 2, 2004, pp. 1026–1031 Vol.2. [Online]. Available: http://dx.doi.org/10.1109/ICNSC.2004.1297088

[17] C. Figuera, J. L. Rojo-Álvarez, M. Wilby, I. Mora-Jiménez, and A. J. Caamaño, "Advanced support vector machines for 802.11 indoor location," *Signal Process.*, vol. 92, no. 9, pp. 2126–2136, Sep. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2012.01.026

[18] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Dec. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1248547.1248632

[19] J. J. Pan, Q. Yang, H. Chang, and D. Y. Yeung, "A manifold regularization approach to calibration reduction for sensor-network based tracking," in *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, Boston, United States, 2006, pp. 988–993.

[20] J. J. Pan and Q. Yang, "Co-localization from labeled and unlabeled data using graph laplacian," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007, pp. 2166–2171.

[21] D. A. Tran and P. Truong, "Total variation regularization for training of indoor location fingerprints," in *ACM MOBICOM Workshop on Mission-Oriented Wireless Sensor Networking (ACM MiseNet 2013)*, Miami, Sep 2013.

[22] D. A. Tran and T. Zhang, "Fingerprint-based location tracking with hodrick-prescott filtering," in *7th IFIP Wireless and Mobile Networking Conference (WMNC 2014)*, Algarve, Portugal, May 2014.

[23] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 425–437, Feb 2002.

[24] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, "Incremental clustering and dynamic information retrieval," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, ser. STOC '97. New York, NY, USA: ACM, 1997, pp. 626–635.