Regularization-based Location Fingerprinting

Duc A. Tran

Abstract Location fingerprinting is an approach to GPS-free localization. This approach requires a prior training set of fingerprints sampled at known locations, by comparing to which the locations of future fingerprints can be determined. For good accuracy, the training set should be large enough to appropriately cover the area. However, in practice, a quality training set is not easy to obtain and as such recent studies have resorted to utilizing fingerprints that are available without location information; these are called unlabeled fingerprints. This chapter presents several ways one can use regularization to learn from unlabeled fingerprints. Regularization is a mathematical framework to learn a function from data by enforcing regularizers to improve generalizability. The following scenarios are discussed: (1) how the training set can be enriched with unlabeled fingerprints, (2) how a trajectory of a moving device can be computed given its sequential fingerprints, labeled or unlabeled, collected during the trajectory, and (3) how a device can be tracked in an online manner as it moves using real-time fingerprints.

1 Introduction

Location information is valuable to a myriad of applications of wireless networks. In a surveillance sensor network, it is crucial to know the location of an incident caught by a sensor, such as fire in a building or oil spill in a coastal water. The demand is also high for mobile apps providing navigation and other location-based services in hospitals, shopping malls, airport terminals, and campus buildings, to name a few. GPS is the most effective way to get location information but does not work indoors. Even for outdoor environments where this service is available, it is not energy-efficient to have to turn it on continuously all the time.

Duc A. Tran
Department of Computer Science, University of Massachusetts Boston, e-mail: duc.tran@umb.edu

Consequently, numerous efforts have been made towards GPS-free localization solutions. A popular approach is to leverage a model correlating received signal strength (RSS) with distance [30]; this is called "ranging". Given a number of reference points (RPs), e.g., Wi-Fi access points [1] or FM broadcasting towers [8], we can locate a device by estimating its distances to these RPs based on RSS ranging and then using multi-lateration to compute the device's location. RSS ranging, however, is highly sensitive to noise interference [30]. Furthermore, radio propagates differently in different directions due to obstacles such as walls, people, and furniture. Positioning based on LED lighting [12] has also been proposed with promising accuracy, but as visible light does not penetrate physical obstacles, this technique is suitable only for short-range applications.

Location fingerprinting is a viable range-free localization alternative. An early adopter of this approach is RADAR [1], perhaps the world's first Wi-Fi RSS-based indoor positioning system. This system relies on a radio map, a lookup table that maps locations inside the building under localization to the RSS fingerprints empirically observed at these locations, respectively. The reference points are the Wi-Fi access points in the building. To locate a user, the radio map is searched to find the closest sample RSS reading and its corresponding location will be used as the estimate for the user's location. RADAR represents the fingerprint approach that uses kNN for comparison to the map [1, 16, 33]. One can also employ a model-based learning approach to relate a fingerprint to a location, for example, probabilistically using Bayesian inference [21] or non-probabilistically using Artificial Neural Networks [13] or Support Vector Machines [5, 11, 32].

To generalize, a fingerprint at a specific location is a vector of location-sensitive measurements observed about the mobile device at this location. For indoor environments, such a measurement can be a RSS reading from a nearby Wi-Fi access point [1], a FM broadcasting tower [8], or a cellular tower [29]. For underwater environments, a measurement can be a profile of echo-sounding signals transmitted from the device (e.g., an AUV) to the sea floor or ping signals to the surface buoys. In theory, any sensor information that is sensitive to location change, including sound [23], light [12], and geomagnetic field [9], can be included in the fingerprint vector. Combining different sensor data where applicable can lead to a rich set of discriminative features for the fingerprint information.

The fingerprint approach works on the basis that if fingerprint information is obtained for sufficiently many sample locations then the device's location given a new fingerprint can be computed by comparing to these samples. Specifically, there are two phases: the training phase, which is often done offline, and the positioning phase, which is done online. In the training phase, a number of sample locations are surveyed to build a map pairing each location to a fingerprint. In the positioning phase, when we need to compute a location in real time, the fingerprint of the device is compared against the fingerprint map to find the best location match.

Despite its simplicity, the fingerprint approach is limited by the quality of the training data. The training data should be sufficiently large to be well-representative of the environment, both spatially and temporally; see example in Figure 1. For a large area, many locations need to be surveyed to ensure good spatial coverage and

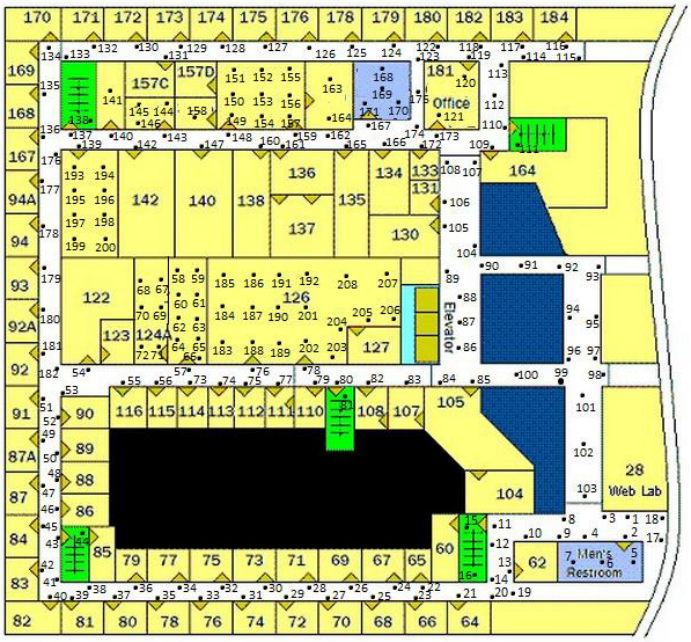


Fig. 1 Example of surveyed area: Wi-Fi RSSI fingerprint data were obtained at 208 sample locations (shown as dots) on the Computer Science department floor (68m $\times$ 63m) at UMass Boston. There are in total 138 Wi-Fi access points and from those unreachable the corresponding RSSI is set to -100db. At each sample location, the corresponding fingerprint is the average of the RSSIs observed at this location. RSSI was measured by a person carrying an Android phone in no particular heading direction.

many fingerprint readings need to be measured at each sample location to ensure good temporal coverage (the signal characteristics of the environment are not time invariant). Consequently, the calibration task can be tedious and labor-extensive, causing bottleneck to localization accuracy.

To circumvent this problem, one can apply semi-supervised learning [6] to augment the (small) training data set of fingerprints with non-training fingerprints (those available but without known location) [17–19, 25, 34]. Here, training fingerprints serve as labeled data and non-training fingerprints as unlabeled. Unlabeled fingerprints are abundant because they can easily be obtained for a mobile device without manual location labeling. In practice, fingerprints should have similar values at similar locations and differ at different locations. This spatial property can be useful to regulate the learning. In Section 3, we present two different ways how location fingerprinting can be cast into a semi-supervised learning framework using regularization.

Another problem discussed in this chapter is location tracking of a mobile device based on its sequentially obtained fingerprints. As the device is moving, more fingerprints, labeled or unlabeled, may be obtained on the spot and we should utilize

them to better locate the device beyond mere reliance on the prebuilt/enriched training data. Since these fingerprints are that of a trajectory, a useful observation is that if two fingerprints are measured in proximate times then so should their corresponding locations. Section 4 presents the formulation of fingerprint-based tracking as a regularization problem incorporating this kind of temporal smoothness to compute a moving device's trajectory from its sequential fingerprint history.

Any computing framework is meaningful only if it can be implemented efficiently in a real system. For fingerprint-based localization and tracking in real time, since the fingerprint stream can go to infinity, it is impossible to store and process all the fingerprints observed in the past due to storage and computation costs. It is therefore desirable to have an algorithm that can compute the location in an online manner using a manageable amount of memory and compute resources. One idea is to store and compute based on only a small set of representative fingerprints instead of all the observed fingerprints. The rationale is as follows. Since the location matrix of a mobile device over a time window exhibits a low-rank structure, as substantiated in [20], we conjecture that the fingerprint matrix over time should also be sparse because of the tight correspondence between a fingerprint and its location. Consequently, the fingerprint stream can be approximated by a sparse set of representative fingerprints with minimal loss of information. Section 5 presents an online algorithm based on this idea.

There are already numerous research works on mobile localization and tracking, but they make additional assumptions such that those about special sensors built in the device (e.g., gyroscope, accelerometer, compass, light sensor) [31], those about mobility-specific constraints (e.g, speed, predefined map) [33] and those that are network-specific (e.g., vehicular [3] or wireless sensor networks [24]). In contrast, the regularization frameworks presented in this chapter are aimed at universal applicability in the sense that they are orthogonally applicable to any type of fingerprint space; i.e., applicable where fingerprint information can be of radio signals, acoustic, visible light, or geomagnetic, etc and can contain any other information so long as it is location-sensitive.

The remainder of this chapter is organized as follows. Section 2 presents some background about fingerprint localization, its state-of-the-art formulation and solution using a training dataset. Section 3 discusses how to enrich the training data with unlabeled fingerprint information. Section 4 is focused on the trajectory reconstruction problem. Section 5 is about how a trajectory can be computed in a sequential manner. Finally, the chapter is concluded in Section 6.

2 Preliminaries

Denote the fingerprint space by $\mathscr{X} \subset \mathbb{R}^m$, where m is the number of fingerprint features, each taking a real-valued number; e.g., RSSI from different Wi-Fi APs, readings from inertial measurement units (accelerometer, gyroscope, magnetometer), and/or any other location-discriminative feature that is obtainable for the de-

vice. A fingerprint is said to be "labeled" if its ground-truth location is known and "unlabeled" otherwise. For a fingerprint $\mathbf{x}$, which is a point in this m-dimensional space, let $h(\mathbf{x})$ be the function indicating whether $\mathbf{x}$ is labeled ($h(\mathbf{x}) = 1$) or not ($h(\mathbf{x}) = 0$). Denote

$$y(\mathbf{x}) = \begin{cases} \text{ground-truth location of } \mathbf{x}, \text{ if } \mathbf{x} \text{ is labeled} \\ 0, \text{ if } \mathbf{x} \text{ is unlabeled.} \end{cases}$$

For the sake of presentation, we assume that the location space is 1D; hence, $y(\mathbf{x}) \in \mathbb{R}$. The case for higher dimensions is a trivial extension (where each coordinate is computed separately). The unknown in our formulation is a function $f : \mathscr{X} \to \mathbb{R}$ that returns the location estimate for a given fingerprint. Ideally, $f(\mathbf{x})$ should equal its ground-truth location $y(\mathbf{x})$ for every labeled $\mathbf{x}$.

Given a training set of labeled fingerprints, $T = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l\}$, one can use supervised learning to learn the location estimate f. For example, supervised learning can be formulated as a Tikhonov regularization problem minimizing the following regularized empirical risk

$$f^* = \underset{f \in \mathscr{H}_K}{\operatorname{argmin}} \sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 + \lambda_K \|f\|_K^2. \tag{1}$$

Here, the solution space for the location estimator f is the reproducing kernel Hilbert space (RKHS) $\mathscr{H}_K$ associated with a kernel function

$$\begin{aligned} K : \mathscr{X} \times \mathscr{X} &\to \mathbb{R} \\ (\mathbf{x}, \mathbf{x}') &\mapsto \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2\gamma^2}\right) \end{aligned} \tag{2}$$

for some constant γ. The first term, $\sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2$, represents the deviation compared to the ground truth, using the squared loss as an example. The second term, $\lambda_K \|f\|_K^2$, is added to enforce some property; in our case, f is preferred to be smooth with respect to kernel K. The notation $\|.\|_K$ denotes the norm in $\mathscr{H}_K$. Coefficient $\lambda_K > 0$ represents the enforcement weight of the regularization.

In our context, the RKHS $\mathscr{H}_K$ is a Hilbert space of real-valued functions defined on the fingerprint space $\mathscr{X}$, with the following properties. First, for every $\mathbf{x} \in \mathscr{X}$, the function $K_\mathbf{x} \equiv K(\mathbf{x}, .) \in \mathscr{H}_K$. Second, which is referred to as the reproducing property, for every $\mathbf{x} \in \mathscr{X}$, we have $f(\mathbf{x}) = \langle f, K_\mathbf{x} \rangle$. In general, any symmetric positive definite function can be used for the kernel K, not just the Gaussian function defined in Eq. 2.

Because $\mathscr{H}_K$ is a vector space whose dimension can be infinite, trying to directly solve the minimization problem (1) is not computationally feasible. Fortunately, thanks to the beautiful theorem below, called the Representer Theorem [22], we can convert this problem to a minimization problem in a finite-dimensional space, which can be solved easily.

Theorem 1 (Representer Theorem). *Suppose we are given a non-empty set $\mathscr{X}$, a positive definite kernel $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$, a set of training samples $\{(\mathbf{x}_1, y(\mathbf{x}_1)), (\mathbf{x}_2, y(\mathbf{x}_2)), ..., (\mathbf{x}_l, y(\mathbf{x}_l))\}$, a strictly increasing function $g : [0,\infty) \to \mathbb{R}$, an arbitrary cost function $c : (\mathscr{X} \times \mathbb{R}^2)^l \to \mathbb{R} \cup \{\infty\}$. Then any $f \in \mathscr{H}_K$ minimizing the regularized risk functional*

$$c\left((\mathbf{x}_1, y(\mathbf{x}_1), f(\mathbf{x}_1)), (\mathbf{x}_2, y(\mathbf{x}_2), f(\mathbf{x}_2)), ..., (\mathbf{x}_l, y(\mathbf{x}_l), f(\mathbf{x}_l))\right) + g\left(\|f\|_K\right) \tag{3}$$

must lie in the subspace spanned by $\{K_{\mathbf{x}_1}, K_{\mathbf{x}_2}, ..., K_{\mathbf{x}_l}\}$.

Proof. Project f onto the subspace spanned by the vectors $\{K_{\mathbf{x}_1}, K_{\mathbf{x}_2}, ..., K_{\mathbf{x}_l}\}$ to obtain

$$f = \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} + v$$

where $v \in \mathscr{H}_K$ is the orthogonal component; i.e,. $\langle v, K_{\mathbf{x}_i} \rangle = 0 \; \forall i = 1, 2, ..., l$. Because of the reproducing property,

$$\begin{aligned} f(\mathbf{x}_j) &= \langle f, K_{\mathbf{x}_j} \rangle \\ &= \left\langle \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} + v, K_{\mathbf{x}_j} \right\rangle \\ &= \sum_{i=1}^{l} \alpha_i \langle K_{\mathbf{x}_i}, K_{\mathbf{x}_j} \rangle + \langle v, K_{\mathbf{x}_j} \rangle \\ &= \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

which is independent from v. Thus, the cost function $c(.)$ does not depend on v.

Next, we have

$$\begin{aligned} \|f\|_K^2 &= \left\langle \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} + v, \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} + v \right\rangle \\ &= \left\langle \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i}, \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} \right\rangle + \langle v, v \rangle \\ &= \left\| \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} \right\|_K^2 + \|v\|_K^2 \end{aligned}$$

and so, as g is strictly increasing,

$$g(\|f\|) \geq g\left(\sqrt{\| \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} \|_K^2} \right);$$

the equality holds when $v = 0$. It becomes obvious that for f to minimize the risk in (3) we must have $v = 0$. It follows that f must lie in the subspace spanned by $\{K_{\mathbf{x}_1}, K_{\mathbf{x}_2}, ..., K_{\mathbf{x}_l}\}$.

Applying this theorem to our problem, where

$$c((\mathbf{x}_1, y(\mathbf{x}_1), f(\mathbf{x}_1)), (\mathbf{x}_2, y(\mathbf{x}_2), f(\mathbf{x}_2)), ..., (\mathbf{x}_l, y(\mathbf{x}_l), f(\mathbf{x}_l))) = \sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2$$

and $g(x) = \lambda_K x^2$ (a strictly increasing function on $[0, \infty)$), hence

$$g(\|f\|_K) = \lambda_K \|f\|_K^2,$$

we will look for a solution of the form

$$f = \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i}$$

or

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \; \forall \mathbf{x} \in \mathscr{X}$$

where the coefficients $\alpha_1, \alpha_2, ..., \alpha_l \in \mathbb{R}$ are the only unknown to be found. Let us denote the following matrices:

- The location estimator vector

$$\mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ ... \\ f(\mathbf{x}_l) \end{bmatrix}$$

- The kernel coefficient vector

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ ... \\ \alpha_l \end{bmatrix}$$

- The label vector

$$\mathbf{y} = \begin{bmatrix} y(\mathbf{x}_1) \\ y(\mathbf{x}_2) \\ ... \\ y(\mathbf{x}_l) \end{bmatrix}$$

- The identity matrix $\mathbf{I} = diag(\underbrace{1, 1, ..., 1}_{l})$
- The kernel matrix

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1)\ K(\mathbf{x}_1,\mathbf{x}_2)\ \ldots\ K(\mathbf{x}_1,\mathbf{x}_l) \\ K(\mathbf{x}_2,\mathbf{x}_1)\ K(\mathbf{x}_2,\mathbf{x}_2)\ \ldots\ K(\mathbf{x}_2,\mathbf{x}_l) \\ \quad\ldots \\ K(\mathbf{x}_l,\mathbf{x}_1)\ K(\mathbf{x}_l,\mathbf{x}_2)\ \ldots\ K(\mathbf{x}_l,\mathbf{x}_l) \end{bmatrix}$$

We have

$$\sum_{i=1}^{l}(f(\mathbf{x}_i)-y(\mathbf{x}_i))^2 = (\mathbf{f}-\mathbf{y})^\intercal(\mathbf{f}-\mathbf{y}) = (\mathbf{K}\boldsymbol{\alpha}-\mathbf{y})^\intercal(\mathbf{K}\boldsymbol{\alpha}-\mathbf{y})$$

and

$$\begin{aligned} \|f\|_K^2 &= \langle f, f\rangle \\ &= \left\langle \sum_{i=1}^{l}\alpha_i K_{\mathbf{x}_i}, \sum_{i=1}^{l}\alpha_i K_{\mathbf{x}_i} \right\rangle \\ &= \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j\langle K_{\mathbf{x}_i}, K_{\mathbf{x}_j}\rangle \\ &= \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x}_j) \\ &= \boldsymbol{\alpha}^\intercal\mathbf{K}\boldsymbol{\alpha} \end{aligned}$$

The minimization problem in (1) can be expressed in matrix form as

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\ \ (\mathbf{K}\boldsymbol{\alpha}-\mathbf{y})^\intercal(\mathbf{K}\boldsymbol{\alpha}-\mathbf{y}) + \lambda_K\boldsymbol{\alpha}^\intercal\mathbf{K}\boldsymbol{\alpha}.$$

Assuming $\mathbf{K}$ is invertible, using gradient descent to solve this minimization problem, we can easily obtain

$$\boldsymbol{\alpha}^* = \begin{bmatrix} \alpha_1^* \\ \alpha_2^* \\ \ldots \\ \alpha_l^* \end{bmatrix} = (\lambda_K\mathbf{I}+\mathbf{K})^{-1}\mathbf{y}.$$

In the positioning phase, given a new fingerprint $\mathbf{x}$, its predicted location will be

$$f^*(\mathbf{x}) = \sum_{i=1}^{l}\alpha_i^* K(\mathbf{x}_i,\mathbf{x}).$$

The localization error of this prediction depends on the quality of the training set T.

3 Enrichment of Training Data

Suppose that, in addition to the original training set $T = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l\}$, whose ground-truth location is known, we have a supplemental set of unlabeled fingerprints, $U = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, ..., \mathbf{x}_{l+u}\}$, whose location is unknown. In practice, $u \gg l$. If we can somehow learn the locations of the unlabeled fingerprints, the extended fingerprint map $\{y(\mathbf{x})\}_{\mathbf{x}\in T\cup U}$ with $n = l + u$ samples, instead of the original map $\{y(\mathbf{x})\}_{\mathbf{x}\in T}$ with only l samples, should be used for training purposes. Since the extended map is richer, it is highly likely that the localization quality during the positioning phase will be better.

One way to learn the locations of these unlabeled fingerprints based on the labeled is by applying Bayesian inference [14]. This method, called the generative method, consists in two steps. First, we estimate a distribution for the probability $p(\mathbf{x}|y)$ of observing a fingerprint $\mathbf{x}$ at a given location y. Second, the location corresponding to a fingerprint $\mathbf{x}$ is determined based on the probability

$$p(y|\mathbf{x}) \propto p(\mathbf{x}|y)p(y)$$

where $p(y)$ is the probability of location y. The location distribution is assumed known.

The generative method relies on choosing the right model for the distribution $p(\mathbf{x}|y)$, which is not easy. Alternatively, one can use the regularization method to propagate the location labels for the unlabeled fingerprints based on their similarity with the labeled. This is possible because of the spatial smoothness in the fingerprint space. The de facto regularization framework for semi-supervised learning is Manifold Regularization proposed by Belkin et al. [2]. Pan et al. [17, 18] apply this framework to fingerprint localization, in which a Laplacian regularization term is added to regulate 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. Total Variation Regularization, which is an alternative framework for semi-supervised learning [15], has been explored for location fingerprinting in the work of Tran and Truong [25]. We present below the formulations using Manifold Regularization and Total Variation Regularization.

3.1 Manifold Regularization

The Manifold Regularization method for semi-supervised learning extends the regularization formulation (1) by introducing a Laplacian regularization term to enforce the smoothness with respect to an intrinsic manifold. To apply this framework to location fingerprinting, as in [17, 18], we need to solve the following problem:

$$\min_{f\in\mathscr{H}_K} \sum_{i=1}^{l}(f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 + \lambda_K\|f\|_K^2 + \lambda_{MR}\sum_{i,j=1}^{n} w(\mathbf{x}_i,\mathbf{x}_j)(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2. \quad (4)$$

Here, the intrinsic manifold is a similarity graph W of $n = l + u$ vertices, each representing a fingerprint. W can be constructed as a kNN-graph or an ε-ball-graph. As a kNN-graph, each vertex is connected to its k nearest vertices; i.e., those at smallest distances (distance in the fingerprint space). As an ε-ball-graph, each vertex is connected to every vertex within a distance bounded by ε. The distance can be defined based on any metric such as Euclidean or Manhattan and the value k or ε chosen must ensure that the graph is connected. Once the edges are formed, each edge $(\mathbf{x}_i, \mathbf{x}_j)$ is associated with a weight $w(\mathbf{x}_i, \mathbf{x}_j)$ to represent the similarity between the fingerprints $\mathbf{x}_i$ and $\mathbf{x}_j$. The similarity measure is most often based on a Gaussian Radial Basis Function; i.e., $w(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{2\sigma^2}\right)$ for some σ. The weight is set to zero for non-edges.

The additional regularization term in (4) is called the Laplacian regularizer because we can express

$$\sum_{i,j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j)(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \dots \\ f(\mathbf{x}_n) \end{bmatrix} \mathbf{L} \begin{bmatrix} f(\mathbf{x}_1) & f(\mathbf{x}_2) & \dots & f(\mathbf{x}_n) \end{bmatrix}$$

where $\mathbf{L}$ is the Laplacian matrix of the similarity graph W,

$$\mathbf{L} = \left[l_{ij} = \begin{cases} -w(\mathbf{x}_i, \mathbf{x}_j) & \text{if } i \neq j \\ \sum_{k=1}^{n} w(\mathbf{x}_i, \mathbf{x}_k) & \text{otherwise} \end{cases} \right]_{n \times n}$$

Let us denote $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_n)]^\intercal$, $\mathbf{y} = [y(\mathbf{x}_1), y(\mathbf{x}_2), ..., y(\mathbf{x}_n)]^\intercal$, $\mathbf{H} = diag\,(h(\mathbf{x}_1), h(\mathbf{x}_2), ..., h(\mathbf{x}_n))$, the identity matrix $\mathbf{I} = diag(\underbrace{1, 1, ..., 1}_{n})$, and the kernel matrix $\mathbf{K} = [k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)]_{n \times n}$. Then, we can write

$$\sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 = (\mathbf{f} - \mathbf{y})^\intercal \mathbf{H} (\mathbf{f} - \mathbf{y}).$$

Because we just need to find the best values for $\{f(\mathbf{x}_{l+1}), f(\mathbf{x}_{l+2}), ..., f(\mathbf{x}_n)\}$, it suffices to solve the following minimization:

$$\min_{\mathbf{f}} \left\{ J(\mathbf{f}) = (\mathbf{f} - \mathbf{y})^\intercal \mathbf{H} (\mathbf{f} - \mathbf{y}) + \lambda_K \|\mathbf{f}\|_K^2 + \lambda_{MR} \mathbf{f}^\intercal \mathbf{L} \mathbf{f} \right\}. \tag{5}$$

Proposition 1. *The minimizer of risk $J(\mathbf{f})$ in (5) admits the following solution:*

$$\mathbf{f}^* = (\lambda_K \mathbf{I} + \mathbf{K}\mathbf{H} + \lambda_{MR} \mathbf{K}\mathbf{L})^{-1} \mathbf{K}\mathbf{H}\mathbf{y}. \tag{6}$$

Proof. Because f belongs to the RKHS $\mathscr{H}_K$, according to the Representer Theorem, we look for a solution in the form $\mathbf{f} = \mathbf{K}\boldsymbol{\alpha}$ for some column vector $\boldsymbol{\alpha} \in \mathbb{R}^n$. There-

fore, $\|\mathbf{f}\|_K^2 = \boldsymbol{\alpha}^\intercal \mathbf{K}\boldsymbol{\alpha}$ and the risk $J(\mathbf{f})$ in Eq. (5) becomes: (note that $\mathbf{K}$ is symmetric and so $\mathbf{K}^\intercal = \mathbf{K}$)

$$\begin{aligned} J(\mathbf{f}) &= (\mathbf{f}-\mathbf{y})^\intercal \mathbf{H}(\mathbf{f}-\mathbf{y}) + \lambda_K \boldsymbol{\alpha}^\intercal \mathbf{K}\boldsymbol{\alpha} + \lambda_{MR}\mathbf{f}^\intercal \mathbf{L}\mathbf{f} \\ &= \mathbf{f}^\intercal(\mathbf{H}+\lambda_{MR}\mathbf{L})\mathbf{f} - 2\mathbf{y}^\intercal \mathbf{H}\mathbf{f} + \mathbf{y}^\intercal \mathbf{H}\mathbf{y} + \lambda_K \boldsymbol{\alpha}^\intercal \mathbf{K}\boldsymbol{\alpha} \\ &= \boldsymbol{\alpha}^\intercal \mathbf{K}(\mathbf{H}+\lambda_{MR}\mathbf{L})\mathbf{K}\boldsymbol{\alpha} - 2\mathbf{y}^\intercal \mathbf{H}\mathbf{K}\boldsymbol{\alpha} + \mathbf{y}^\intercal \mathbf{H}\mathbf{y} + \lambda_K \boldsymbol{\alpha}^\intercal \mathbf{K}\boldsymbol{\alpha} \\ &= \boldsymbol{\alpha}^\intercal \underbrace{\mathbf{K}\left(\lambda_K \mathbf{I} + (\mathbf{H}+\lambda_{MR}\mathbf{L})\mathbf{K}\right)}_{\mathbf{Q}} \boldsymbol{\alpha} - 2\mathbf{y}^\intercal \mathbf{H}\mathbf{K}\boldsymbol{\alpha} + \mathbf{y}^\intercal \mathbf{H}\mathbf{y} \\ &= \boldsymbol{\alpha}^\intercal \mathbf{Q}\boldsymbol{\alpha} - 2\mathbf{y}^\intercal \mathbf{H}\mathbf{K}\boldsymbol{\alpha} + \mathbf{y}^\intercal \mathbf{H}\mathbf{y}. \end{aligned}$$

To minimize $J(\mathbf{f})$, set its derivative with respect to $\boldsymbol{\alpha}$ to zero,

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = (\mathbf{Q}+\mathbf{Q}^\intercal)\boldsymbol{\alpha} - 2\mathbf{K}\mathbf{H}\mathbf{y} = 0.$$

Since $\mathbf{K}$, $\mathbf{H}$, and $\mathbf{L}$ are symmetric, we have $\mathbf{Q}^\intercal = \mathbf{Q}$ and so $2\mathbf{Q}\boldsymbol{\alpha} - 2\mathbf{K}\mathbf{H}\mathbf{y} = 0$ or $\left(\lambda_K \mathbf{I} + \mathbf{K}(\mathbf{H}+\lambda_{MR}\mathbf{L})\right)\mathbf{K}\boldsymbol{\alpha} - \mathbf{K}\mathbf{H}\mathbf{y} = 0$. Because $\mathbf{f} = \mathbf{K}\boldsymbol{\alpha}$, we obtain

$$\mathbf{f} = \left(\lambda_K \mathbf{I} + \mathbf{K}\mathbf{H} + \lambda_{MR}\mathbf{K}\mathbf{L}\right)^{-1} \mathbf{K}\mathbf{H}\mathbf{y},$$

assuming the matrix inverse is possible.

3.2 Total Variation Regularization

Total Variation (TV) regularization is a widely used regularization framework for restoring images in the area of image processing [15]. TV permits sharper edges near the decision boundaries whereas the Laplacian regularization penalizes too much gradients on edges. The former's effectiveness as an alternative framework for semi-supervised learning has been demonstrated, for example by Bresson and Zhang [4]. TV can be used to enrich the training quality for location fingerprinting, as first reported in the work of Tran and Truong [25].

In the TV framework for semi-supervised learning, the minimization problem is

$$\min_f \sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 + \lambda_{TV} TV_W, \tag{7}$$

where TV_W is the TV of the function f on the similarity graph W described earlier. By definition, the TV of a continuous function f is

$$TV[f] = \int_{\mathscr{X}} \|\nabla f\| \, dx,$$

where $\mathscr{X}$ is the domain (continuous) of f, ∇f is its gradient, and dx the area element of Ω of f. This concept can be extended for weighted graphs as follows. On graph

W, the TV of a real-valued scalar function f defined on its vertices is the sum of the local TV at each and every vertex,

$$TV_W = \sum_{i=1}^{n} \|\nabla f(i)\|_{L^p(w)}.$$

The local TV at vertex $\mathbf{x}_i$ is the weighted L^p-norm of the gradient at this vertex. The gradient of function f at vertex $\mathbf{x}_i$ is

$$\nabla f(\mathbf{x}_i) = \begin{pmatrix} f(\mathbf{x}_1) - f(\mathbf{x}_i) \\ f(\mathbf{x}_2) - f(\mathbf{x}_i) \\ \cdots \\ f(\mathbf{x}_j) - f(\mathbf{x}_i) \\ \cdots \\ f(\mathbf{x}_n) - f(\mathbf{x}_i) \end{pmatrix}$$

and so,

$$\|\nabla f(\mathbf{x}_i)\|_{L^p(w)} = \left(\sum_{j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j) |f(\mathbf{x}_j) - f(\mathbf{x}_i)|^p \right)^{1/p}.$$

The graph TV above is a generalization of that defined in [4] and [10]. Note that the case $p = 1$ corresponds to the graph TV used by Bresson and Zhang [4],

$$TV_W = \sum_{i,j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j) |f(\mathbf{x}_j) - f(\mathbf{x}_i)|,$$

and the case $p = 2$ corresponds to the graph TV used by Elmoataz et al. [10],

$$TV_W = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j) (f(\mathbf{x}_j) - f(\mathbf{x}_i))^2}.$$

The minimization in (7) becomes

$$\min_f \sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 + \lambda_{TV} \left(\sum_{j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j) |f(\mathbf{x}_j) - f(\mathbf{x}_i)|^p \right)^{1/p}. \quad (8)$$

This problem can be solved generally using the algorithm in [10]. Alternatively, there is a simpler algorithm [25] which fixes the locations for the labeled (setting $f(\mathbf{x}_i) = y(\mathbf{x}_i)$ for labeled $\mathbf{x}_i$) and iteratively adjust the location estimates for the unlabeled as long as the value of the TV regularization term continues to decrease. This algorithm works as follows:

1. Initial step: for $i, j \in [1, n]$

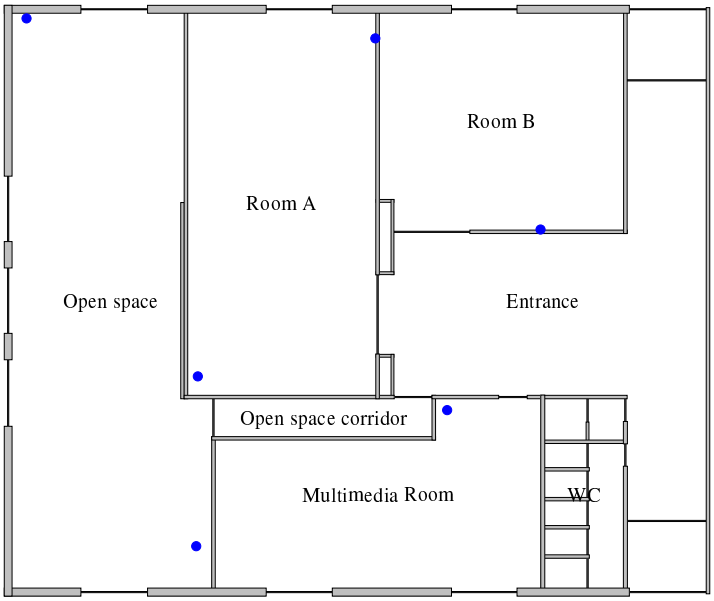


Fig. 2 Trento dataset's map: 30m×20m (courtesy of [5])

$$f_i^{(0)} = \begin{cases} y(\mathbf{x}_i) & \text{if } i \le l \\ 0 & \text{otherwise} \end{cases}$$
$$\gamma_{ij}^{(0)} = w(\mathbf{x}_i, \mathbf{x}_j)$$

2. Iterative step $t = 0, 1, 2, \ldots$: for $i, j \in [1, n]$

$$f_i^{(t+1)} = \begin{cases} f_i^{(t)} & \text{if } i \le l \\ \frac{\sum_{j=1}^n \gamma_{ij}^{(t)} f_j^{(t)}}{\sum_{j=1}^n \gamma_{ij}^{(t)}} & \text{otherwise} \end{cases}$$
$$\gamma_{ij}^{(t+1)} = \frac{w(\mathbf{x}_i, \mathbf{x}_j)}{\|\nabla f^{(t)}(i)\|_{L^2(w)}} + \frac{w(\mathbf{x}_i, \mathbf{x}_j)}{\|\nabla f^{(t)}(j)\|_{L^2(w)}}$$

3. Stop when $\sum_{i=l+1}^n |f_i^{(t+1)} - f_i^{(t)}| < \tau$ (a predefined threshold). When stopped, the value of $f_i^{(t)}$ is used as the estimated location for each unlabeled fingerprint $\mathbf{x}_i$.

3.3 Manifold vs. Total Variation Regularization

We present below the results of an evaluation to assess the effectiveness of Manifold Regularization and Total Variation Regularization for enriching the training

fingerprint set. This evaluation, published in [25], uses a Wi-Fi fingerprint dataset obtained from an indoor experiment, courtesy of [5] (University of Trento), containing 257 RSSI fingerprints at 257 sample locations in a WLAN with six Wi-Fi access points (see floor plan in Figure 2). The sample locations are regular-grid points of the floor. Each fingerprint is measured at a sample location by a person carrying a PDA to receive Wi-Fi signals from the access points. The PDA always points north.

A random half *Train* of this collection (128 samples) is used for training and the other half *Test* (129 samples) for testing purposes. Out of the training samples, we randomly create two groups of samples: the labeled group T (with the location labels intact) and the unlabeled group U (with the location labels removed). It is noted that $T, U \subset Train$ and $T \cap U = \emptyset$. The size of T is set to be 10%, 20%, ..., or 70% of $|Train|$ and, given T, the size of U is set to be 10%, 20%, ..., or 100% of $|Train \setminus T|$. For each combination of these sizes, the average location when tested with *Test* is averaged over 10 times running the simulation (with random generations of T and U). 1-NN is used for testing; that is, given a test fingerprint, its estimated location is the location of the nearest fingerprint in the fingerprint map.

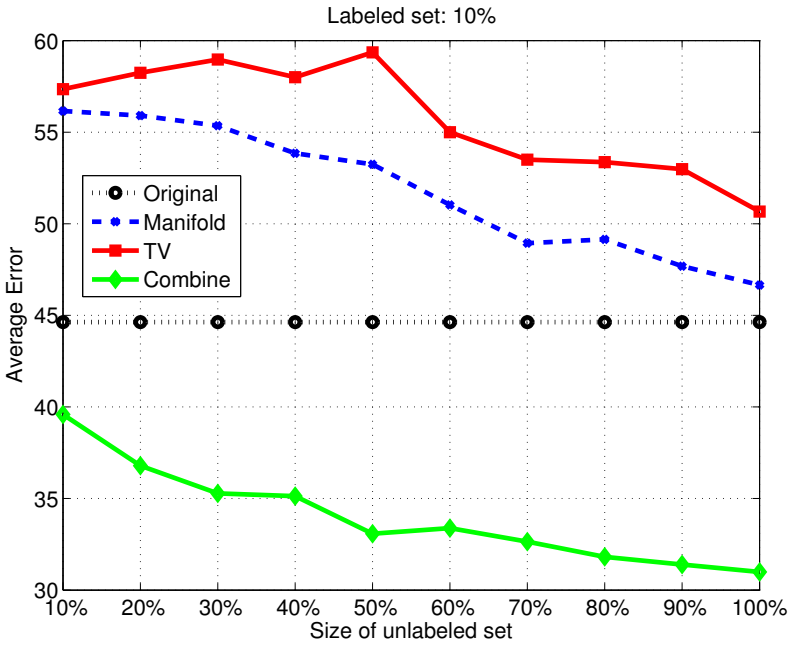
Figure 3 and Figure 4 plot the average error for various cases of $|T|$ and $|U|$, comparing the following techniques.

- Original: only the labeled set T is used as the training fingerprint map.
- Combine: the set $T \cup U$, where the ground-truth label is provided for every fingerprint, is used as the training fingerprint map.
- TV: the Total Variable Regularization method is used to estimate the label for U and then $T \cup U$ is used for training.
- Manifold: the Manifold Regularization method is used to estimate the label for U and then $T \cup U$ is used for training.

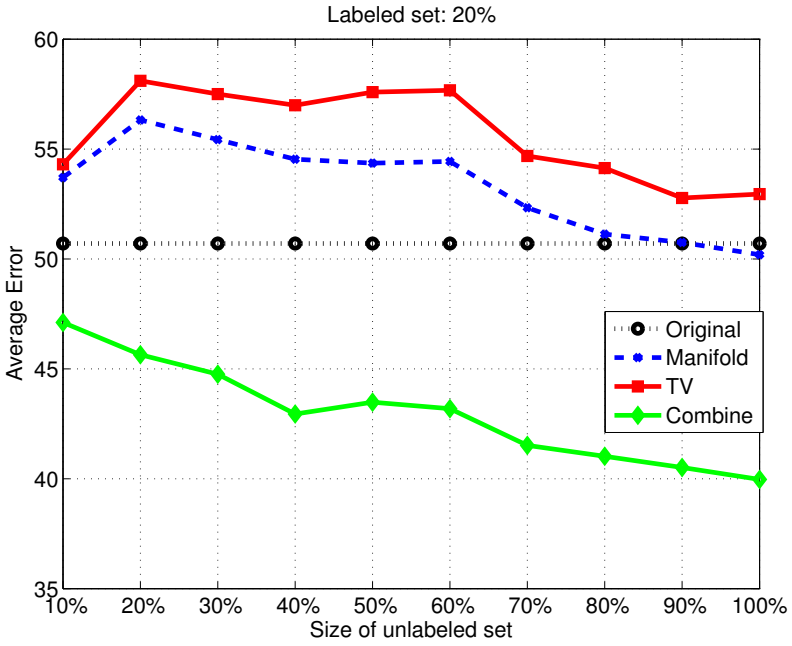
The following patterns are observed:

- Regardless of the size of the labeled set, Manifold and TV tend to be increasingly effective as the size of the unlabeled set increases.
- When the labeled set is small (e.g., 10%, 20% in Figure 3), regularization does not help. Only when the labeled set gets sufficiently large (e.g., 60%, 70% in Figure 4), we start to see its effect. This finding is understandable because a small labeled set offers too little information to be useful for the training.
- Manifold is consistently more accurate than TV. This is different from the observation in the area of image processing where TV is known to be better. This suggests that, unlike images which often have edges, the fingerprint space may not exhibit "edges" of fingerprints (i.e., a path in the fingerprint graph) located at a small cluster of locations isolated from the rest of locations. This could be due to the penetrable-ness of the Wi-Fi signal in the indoor area.

That said, these findings are meant to be suggestive rather than conclusive as the dataset used is small. Nevertheless, they show the potential effectiveness of both Manifold Regularization and TV Regularization in enriching the training data set for fingerprint-based localization.

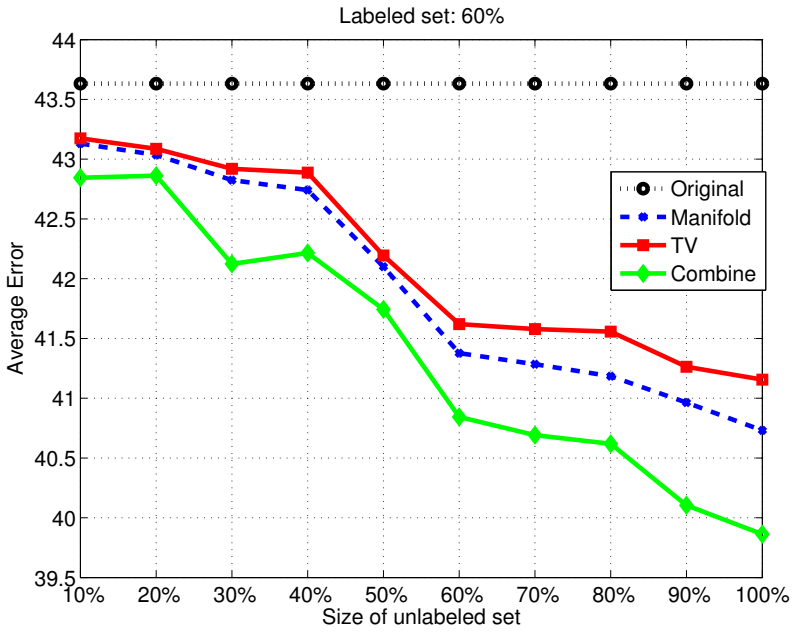


(a) $|T| = 10\%$

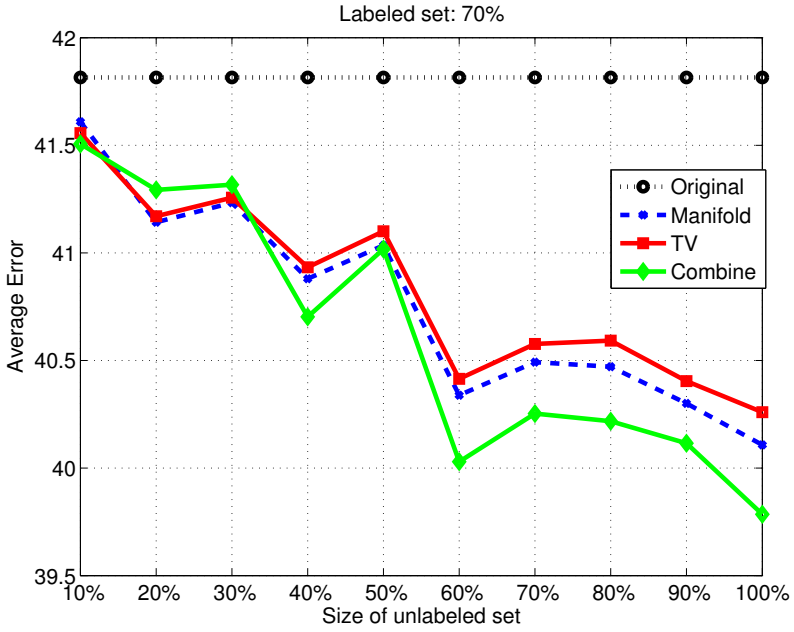


(b) $|T| = 20\%$

Fig. 3 Effectiveness of Manifold Regularization and Total Variation Regularization: showing average location error for the case of low label rate. The unit for y-axis is 0.1m. ([25])



(a) $|T| = 60\%$



(b) $|T| = 70\%$

Fig. 4 Effectiveness of Manifold Regularization and Total Variation Regularization: showing average location error for the case of high label rate. The unit for y-axis is 0.1m. ([25])

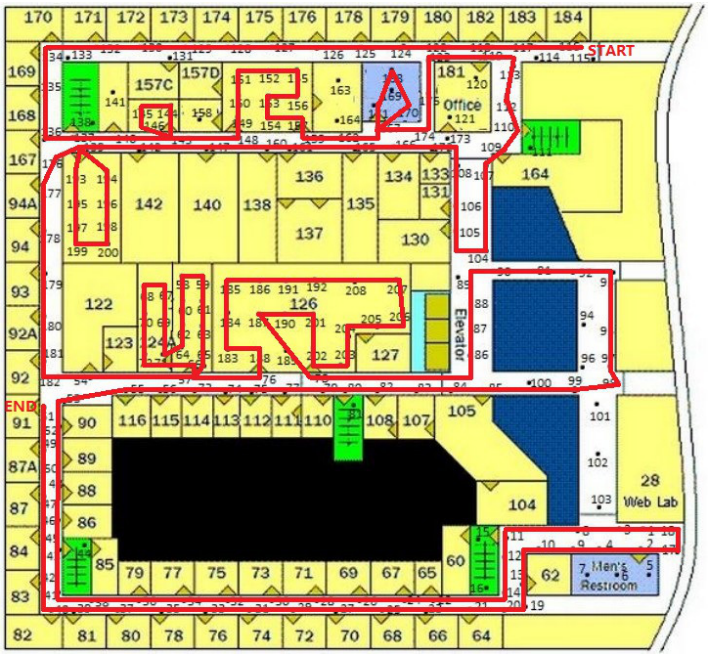


Fig. 5 A test trajectory visiting 185 locations on the Computer Science department floor (68m × 63m) at UMass Boston.

4 Trajectory Computation

Suppose that a mobile device is moving along a trajectory, during which we obtain a sequence of fingerprints, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$, at times $1, 2, ..., n$, respectively. Some of them may be obtained with location information, but most without. Note that this is a sequence of fingerprints whose time order matters, not a set of fingerprints as for enriching the training data in the previous section. We need to compute the location at the current time n.

In a study on trajectory tracking, Rallapalli et al. [20] confirmed that real-world mobility of a device often exhibits its moving at a constant velocity for a long period of time before changing speed. Consequently, the quantity

$$|(f(\mathbf{x}_i) - f(\mathbf{x}_{i-1})) - (f(\mathbf{x}_{i-1}) - f(\mathbf{x}_{i-2}))| = |f(\mathbf{x}_i) + f(\mathbf{x}_{i-2}) - 2f(\mathbf{x}_{i-1})|$$

for most i should be close to zero. We refer to this property as the temporal smoothness in the fingerprint space, in contrast to the spatial smoothness discussed in the previous section.

Tran and Zhang [26] showed that temporal smoothness is more effective than spatial smoothness for trajectory construction if one property is exclusively enforced in the regularization. As an illustration, given the ground-truth trajectory of a mov-

ing device shown in Figure 5, in all three cases where 10% or 50% or 90% of the fingerprint sequence is labeled (at random), a better trajectory estimate is obtained by enforcing temporal smoothness than by enforcing spatial smoothness; see Figure 6, Figure 7, and Figure 8 for the case of 10% labeled, 50%, and 90%, respectively. In these figures, we show the location estimated for a fingerprint instantly at the time it is observed. The first point is always put at the center because in the fingerprint sequence under evaluation it happens to be unlabeled and there is no labeled fingerprint available for learning. As can be seen, in all cases of label rate, the temporally-regularized trajectory resembles the ground-truth trajectory more closely than the spatially-regularized trajectory does. Even in the case only 50% of the fingerprints are labeled, temporal regularization results in a trajectory (Figure 7) comparable to the trajectory produced by spatial regularization for the case 90% labeled (Figure 8).

Tran et. al [28] proposed a unified regularization framework combining both properties as follows:

$$\min_{f\in\mathscr{H}_K}\left\{J(f)=\sum_{i=1}^{n}h(\mathbf{x}_i)(f(\mathbf{x}_i)-y(\mathbf{x}_i))^2+\lambda_K\|f\|_K^2+\lambda_S S(f)+\lambda_T T(f)\right\}, \quad (9)$$

where

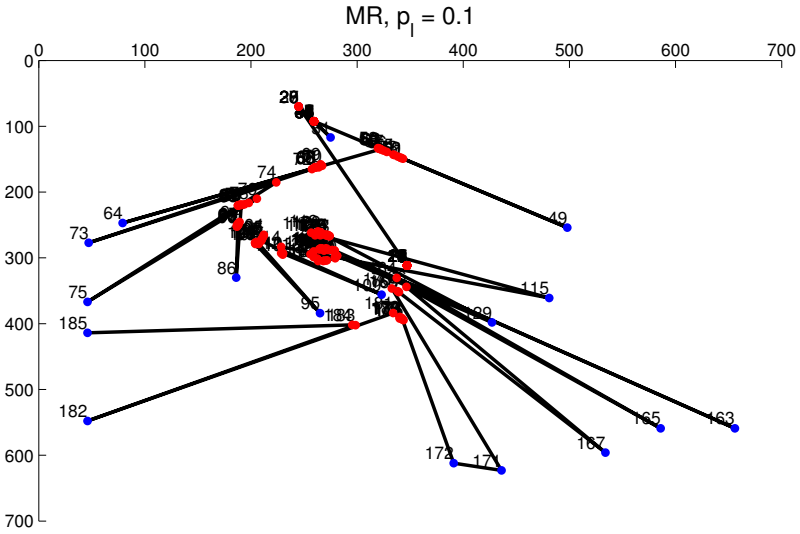
$$S(f)=\sum_{i,j=1}^{n}w(\mathbf{x}_i,\mathbf{x}_j)(f(\mathbf{x}_i)-f(\mathbf{x}_j))^2$$
$$T(f)=\sum_{i=3}^{n}(f(\mathbf{x}_i)+f(\mathbf{x}_{i-2})-2f(\mathbf{x}_{i-1}))^2$$

are the regularizers to enforce spatial smoothness and temporal smoothness, respectively. This framework extends the manifold regularization framework of Belkin et al. [2] (by setting $\lambda_T=0$) with the temporal regularizer $T(f)$. The weights λ_K, λ_S, $\lambda_T\in[0,\infty)$ are to tune the importance of the smoothness terms (kernel, spatial, temporal). Recall that $h(.)$ is the label indicator function.

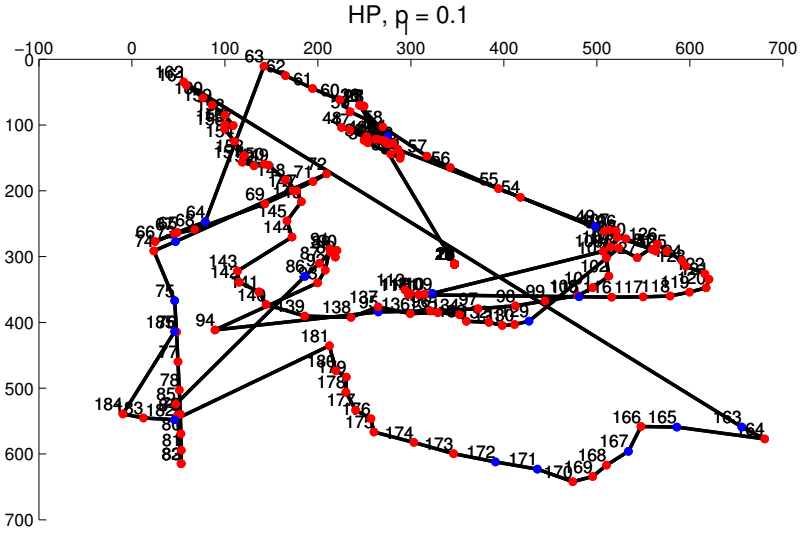
Let $\mathbf{D}$ be the second-order difference matrix of size $n\times n$,

$$\mathbf{D}=\begin{bmatrix}0&0&0&0&\ldots&\ldots&\ldots&\ldots&0\\0&0&0&0&0&\ldots&\ldots&\ldots&0\\1&-2&1&0&0&0&\ldots&\ldots&0\\0&1&-2&1&0&0&0&\ldots&0\\0&0&1&-2&1&0&0&\ldots&0\\\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots&\ldots\\0&\ldots&\ldots&0&1&-2&1&0&0\\0&\ldots&\ldots&\ldots&0&1&-2&1&0\\0&\ldots&\ldots&\ldots&\ldots&0&1&-2&1\end{bmatrix}_{n\times n}$$

Using the same matrix notations, $\mathbf{f}$, $\mathbf{y}$, $\mathbf{H}$, $\mathbf{L}$, and $\mathbf{K}$, as earlier, we have

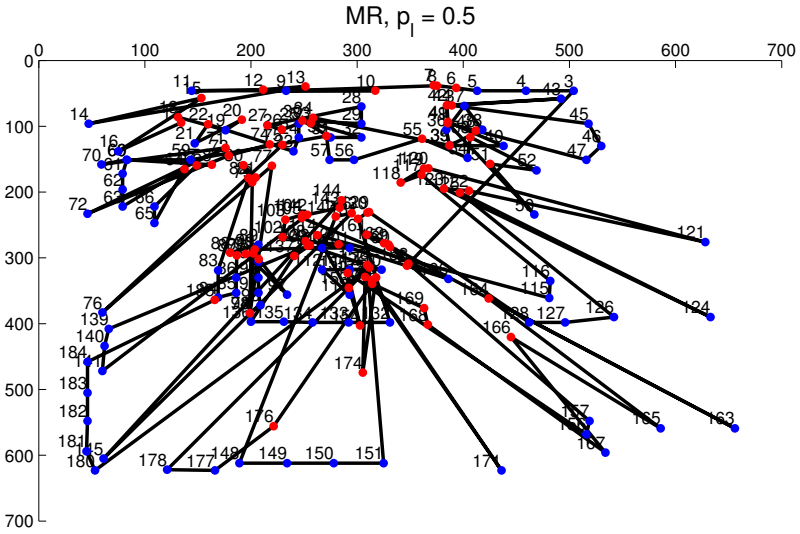


(a) Spatial: 10% labeled

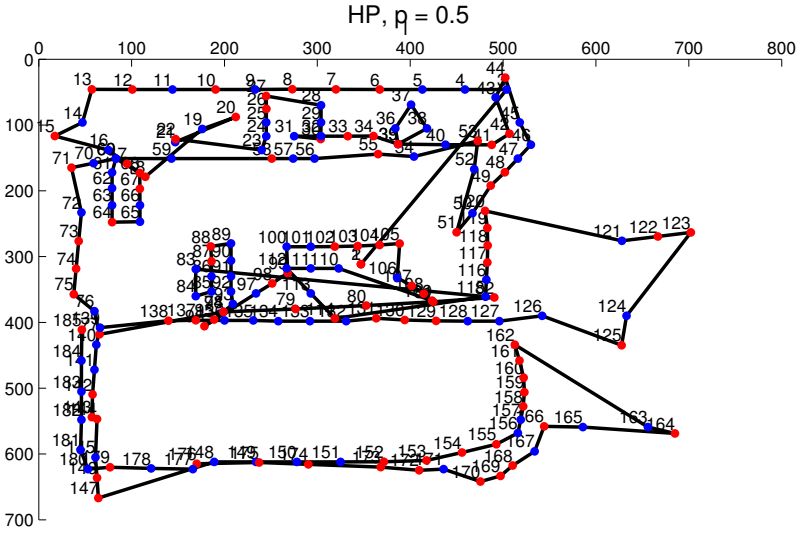


(b) Temporal: 10% labeled

Fig. 6 Effect of spatial regularization vs. temporal regularization for the case 10% of the fingerprints is labeled: showing the estimated trajectory, where 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 measurement. The ground-truth trajectory is shown in Figure 5. ([26])

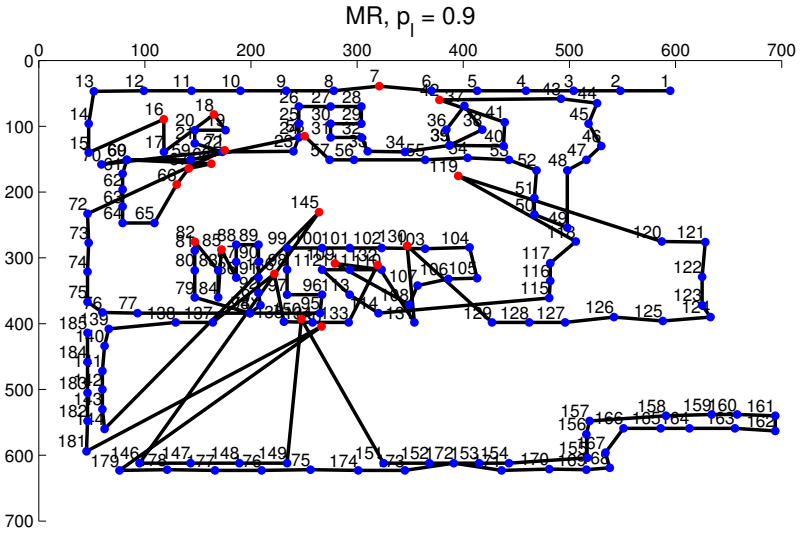


(a) Spatial: 50% labeled

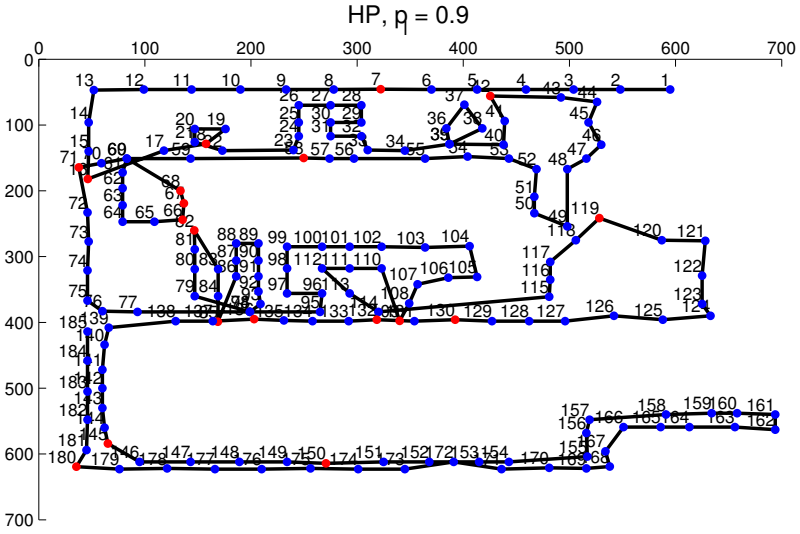


(b) Temporal: 50% labeled

Fig. 7 Effect of spatial regularization vs. temporal regularization for the case 50% of the fingerprints is labeled: showing the estimated trajectory, where 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 measurement. The ground-truth trajectory is shown in Figure 5. ([26])



(a) Spatial: 90% labeled



(b) Temporal: 90% labeled

Fig. 8 Effect of spatial regularization vs. temporal regularization for the case 90% of the fingerprints is labeled: showing the estimated trajectory, where 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 measurement. The ground-truth trajectory is shown in Figure 5. ([26])

$$\mathbf{Df} = \begin{bmatrix} 0 & 0 & 0 & 0 & ... & ... & ... & ... & 0 \\ 0 & 0 & 0 & 0 & 0 & ... & ... & ... & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & ... & ... & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & ... & ... & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & ... & ... & ... & 0 & 1 & -2 & 1 & 0 \\ 0 & ... & ... & ... & ... & 0 & 1 & -2 & 1 \end{bmatrix}_{n\times n} \begin{bmatrix} f(\mathbf{x}_1 \\ f(\mathbf{x}_2 \\ ... \\ f(\mathbf{x}_n) \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ f(\mathbf{x}_1) - 2f(\mathbf{x}_2) + f(\mathbf{x}_3) \\ f(\mathbf{x}_2) - 2f(\mathbf{x}_3 + f(\mathbf{x}_4) \\ ... \\ f(\mathbf{x}_{n-2}) - 2f(\mathbf{x}_{n-1}) + f(\mathbf{x}_n) \end{bmatrix}$$

and so

$$T(f) = \sum_{i=3}^{n} (f(\mathbf{x}_i) + f(\mathbf{x}_{i-2}) - 2f(\mathbf{x}_{i-1}))^2 = (\mathbf{Df})^\intercal \mathbf{Df} = \mathbf{f}^\intercal \mathbf{D}^\intercal \mathbf{Df} = \mathbf{f}^\intercal \mathbf{Bf}$$

where

$$\mathbf{B} = \mathbf{D}^\intercal \mathbf{D} = \begin{bmatrix} 1 & -2 & 1 & 0 & ... & ... & ... & ... & 0 \\ -2 & 5 & -4 & 1 & 0 & ... & ... & ... & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & ... & ... & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 & ... & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & ... & ... & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & ... & ... & ... & 0 & 1 & -4 & 5 & -2 \\ 0 & ... & ... & ... & ... & 0 & 1 & -2 & 1 \end{bmatrix}_{n\times n}.$$

Also, $S(f) = \mathbf{f}^\intercal \mathbf{Lf}$. We now minimize the following regularized risk:

$$\min_{\mathbf{f}} \left\{ J(\mathbf{f}) = (\mathbf{f}-\mathbf{y})^\intercal \mathbf{H}(\mathbf{f}-\mathbf{y}) + \lambda_K \|\mathbf{f}\|_K^2 + \lambda_S \mathbf{f}^\intercal \mathbf{Lf} + \lambda_T \mathbf{f}^\intercal \mathbf{Bf} \right\}. \tag{10}$$

Proposition 2. *The minimizer of risk $J(\mathbf{f})$ in (10) admits the following solution*

$$\mathbf{f}^* = (\lambda_K \mathbf{I} + \mathbf{K}(\mathbf{H} + \lambda_S \mathbf{L} + \lambda_T \mathbf{B}))^{-1} \mathbf{KHy}. \tag{11}$$

Proof. A proof can be derived in a way similar to the proof of Proposition 1. The complete proof is provided in [28].

A comparison of this framework to the nearest-neighbor (NN) framework has been conducted in [28]. For example, for the same ground-truth trajectory in Figure 5, using a sequence of 185 fingerprints with 10% labeled, the corresponding estimate resulted from each framework is visualized in Figure 9; the regularization framework is the better. Figure 10 gives the location error comparison for other cases of label rates.

5 Online Algorithm

Assuming the problem setting in the previous section and using the same notations, we now discuss how the location of a moving device can be computed from its sequentially obtained fingerprints in an online manner. Note that to compute the solution $\mathbf{f}^*$ as in Eq. (11) requires a $O(n^3)$-time and $O(n^2)$-space algorithm. As such, when the fingerprint stream goes larger, it will become infeasible to compute the location of the device in real time. Fortunately, as aforementioned, the fingerprint space should be sparse in both time and space. This implies the possibility to approximate the growing set of fingerprints with only a sparse representation which we can use to estimate location in real time.

5.1 Sparse Representation

Let us represent the set of fingerprints $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ compactly as a multi-set

$$\{(\bar{\mathbf{x}}_1, m_1), (\bar{\mathbf{x}}_2, m_2), ..., (\bar{\mathbf{x}}_k, m_k)\}$$

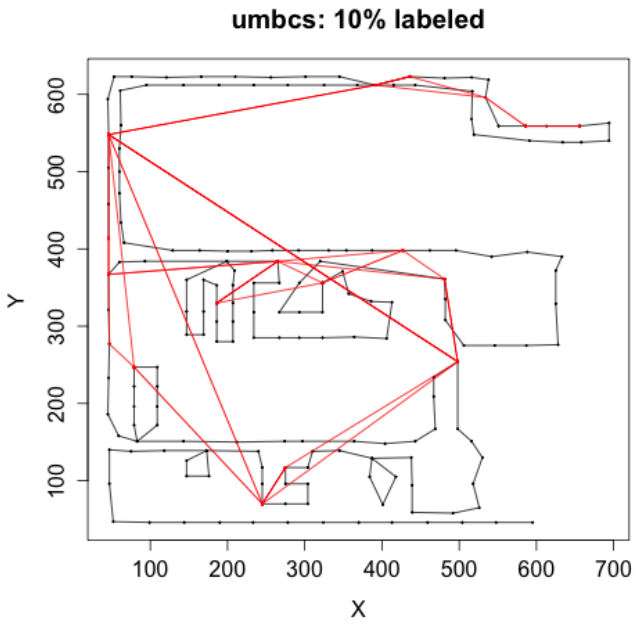
where the representative elements are $\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, ..., \bar{\mathbf{x}}_k$ and their respective multiplicities $m_1, m_2, ..., m_k$ ($\sum_{i=1}^{k} m_i = n$). In other words, there are m_1 fingerprints with value $\bar{\mathbf{x}}_1$, m_2 fingerprints with value $\bar{\mathbf{x}}_2$, etc. Denote the corresponding compact vector for estimated locations by

$$\bar{\mathbf{f}} = \begin{bmatrix} f(\bar{\mathbf{x}}_1) \\ f(\bar{\mathbf{x}}_2) \\ ... \\ f(\bar{\mathbf{x}}_k) \end{bmatrix}.$$

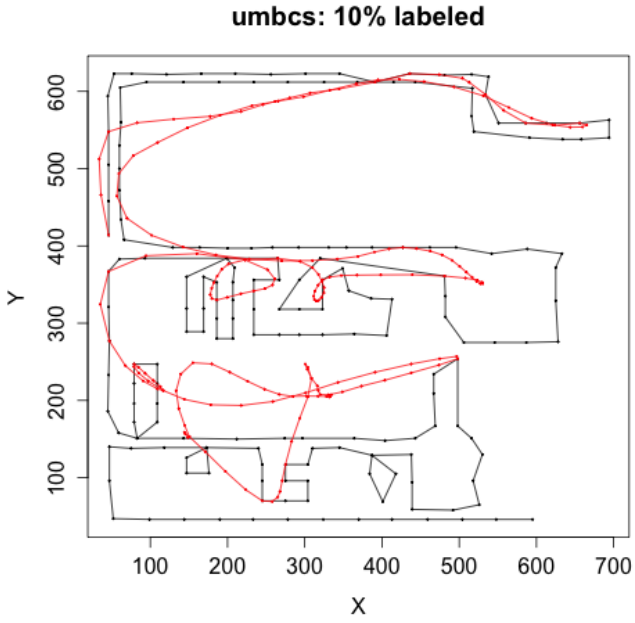
Similarly, let

$$\bar{\mathbf{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ ... \\ \bar{y}_k \end{bmatrix}$$

be the compact vector for ground-truth locations. If no ground-truth information is available for fingerprint $\bar{\mathbf{x}}_i$ and neither for any fingerprint that $\bar{\mathbf{x}}_i$ represents, then



(a) Nearest-neighbor



(b) Regularization

Fig. 9 Regularization vs. Nearest-Neighbor for computation of mobile trajectory: showing the estimated trajectory (red-colored) given the same fingerprint sequence with 10% labeled; the ground-truth trajectory is black-colored. The ground-truth trajectory is shown in Figure 5. ([28])

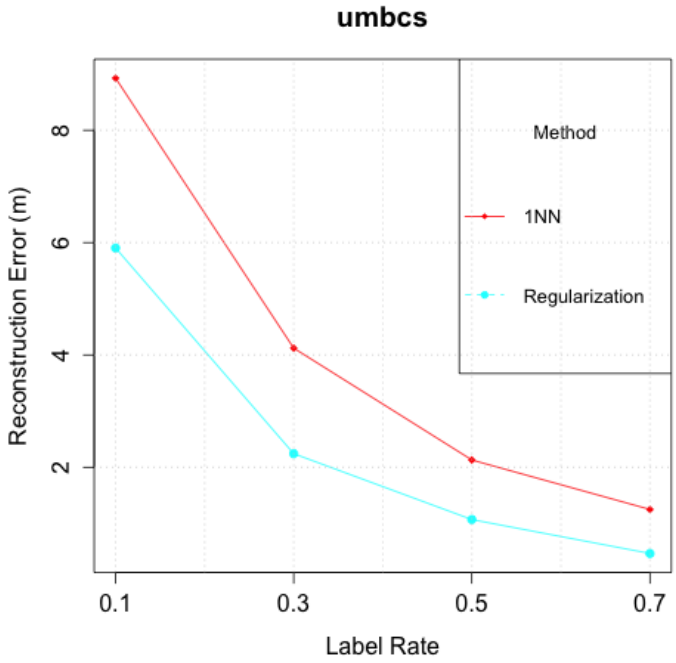


Fig. 10 Regularization vs. Nearest-Neighbor for computation of mobile trajectory: showing location error as more fingerprints are available as labeled.

$\bar{y}_i$ is set to 0; otherwise, $\bar{y}_i$ is the ground-truth location associated with $\bar{\mathbf{x}}_i$. We can transform the original minimization problem in (9) to a more compact version using only the representative fingerprints.

Since we are looking for $f \in \mathscr{H}_K$, let $f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x})$. We have

$$\begin{aligned} f(\bar{\mathbf{x}}_i) &= \sum_{j=1}^{n} \alpha_j K(\mathbf{x}_j, \bar{\mathbf{x}}_i) \\ &= \sum_{j=1}^{k} K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_i) \underbrace{\sum_{p|\mathbf{x_p}=\bar{\mathbf{x}}_j} \alpha_p}_{\bar{\alpha}_j} \\ &= \sum_{j=1}^{k} \bar{\alpha}_j K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_i), \end{aligned}$$

and so $\bar{\mathbf{f}} = \bar{\mathbf{K}}\bar{\boldsymbol{\alpha}}$ where $\bar{\mathbf{K}} = \left[\bar{k}_{ij} = K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_i)\right]_{k\times k}$ and $\bar{\boldsymbol{\alpha}} = [\bar{\alpha}_1, \bar{\alpha}_2, ..., \bar{\alpha}_k]^\intercal$. It follows that the kernel regularization term can be expressed as $\|\mathbf{f}\|_K^2 = \bar{\boldsymbol{\alpha}}^\intercal \bar{\mathbf{K}} \bar{\boldsymbol{\alpha}}$.

Next, the estimation error with respect to the labeled fingerprints can be rewritten as follows:

$$\sum_{i=1}^{n} h_i(f(\mathbf{x}_i)-y_i)^2 = \sum_{i=1}^{k} \bar{h}_i(f(\bar{\mathbf{x}}_i)-\bar{y}_i)^2 = (\bar{\mathbf{f}}-\bar{\mathbf{y}})^\intercal \bar{\mathbf{H}}(\bar{\mathbf{f}}-\bar{\mathbf{y}})$$

where $\bar{\mathbf{H}} = diag(\bar{h}_1, \bar{h}_2, ..., \bar{h}_k)$ such that $\bar{h}_i = \sum_{j|\mathbf{x}_j=\bar{\mathbf{x}}_i} h(\mathbf{x}_j)$ which is the number of labeled fingerprints represented by $\bar{\mathbf{x}}_i$.

The spatial regularization term $S(f)$ can be rewritten as

$$\begin{aligned}
S(f) &= \mathbf{f}^\intercal \mathbf{L} \bar{\mathbf{f}} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{i} w(\mathbf{x}_i,\mathbf{x}_j)(f(\mathbf{x}_i)-f(\mathbf{x}_j))^2 \\
&= \sum_{i=1}^{k}\sum_{j=1}^{i} \underbrace{m_i m_j w(\bar{\mathbf{x}}_i,\bar{\mathbf{x}}_j)}_{\bar{w}(\bar{\mathbf{x}}_i,\bar{\mathbf{x}}_j)} (f(\bar{\mathbf{x}}_i)-f(\bar{\mathbf{x}}_j))^2 \\
&= \sum_{i=1}^{k}\sum_{j=1}^{i} \bar{w}(\bar{\mathbf{x}}_i,\bar{\mathbf{x}}_j) (f(\bar{\mathbf{x}}_i)-f(\bar{\mathbf{x}}_j))^2 = \bar{\mathbf{f}}^\intercal \bar{\mathbf{L}} \bar{\mathbf{f}}
\end{aligned}$$

where $\bar{\mathbf{L}}$ is the Laplacian matrix of the similarity graph with vertices $\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, ..., \bar{\mathbf{x}}_k$, constructed as discussed earlier in Section 3.1, except that the edge weight function is $\bar{w}(\bar{\mathbf{x}}_i,\bar{\mathbf{x}}_j) = m_i m_j w(\bar{\mathbf{x}}_i,\bar{\mathbf{x}}_j)$.

The temporal regularization term $T(f)$ can be rewritten as

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

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.

Therefore, the risk $J(\mathbf{f})$ in (10) can be expressed as

$$J(\mathbf{f}) = (\bar{\mathbf{f}}-\bar{\mathbf{y}})^\intercal \bar{\mathbf{H}}(\bar{\mathbf{f}}-\bar{\mathbf{y}}) + \lambda_K \bar{\boldsymbol{\alpha}}^\intercal \bar{\mathbf{K}} \bar{\boldsymbol{\alpha}} + \lambda_S \bar{\mathbf{f}}^\intercal \bar{\mathbf{L}} \bar{\mathbf{f}} + \lambda_T \bar{\mathbf{f}}^\intercal \bar{\mathbf{B}} \bar{\mathbf{f}}.$$

meaning that the minimization problem in (10) can be reduced to an exact same minimization problem except that

- The input fingerprints are $\{\bar{\mathbf{x}}_i\}$ instead of $\{\mathbf{x}_i\}$.
- The weight function is $\bar{w}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = m_i m_j w(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$.
- The matrices $\bar{\mathbf{L}}$, $\bar{\mathbf{H}}$, $\bar{\mathbf{B}}$, and $\bar{\mathbf{K}}$ are used in place of the matrices $\mathbf{L}$, $\mathbf{H}$, $\mathbf{B}$, and $\mathbf{K}$, respectively.

Applying Proposition 2 on this compact problem, we obtain the following result.

Corollary 1. *The minimizer of risk $J(\mathbf{f})$ in (10) admits the following solution $\mathbf{f}^* = [f^*(\mathbf{x}_1), f^*(\mathbf{x}_2), ..., f^*(\mathbf{x}_n)]$ where $f^*(\mathbf{x}_i)$ is the j^{th} entry of vector*

$$\bar{\mathbf{f}}^* = \left(\lambda_K \bar{\mathbf{I}} + \bar{\mathbf{K}}(\bar{\mathbf{H}} + \lambda_S \bar{\mathbf{L}} + \lambda_T \bar{\mathbf{B}})\right)^{-1} \bar{\mathbf{K}} \bar{\mathbf{H}} \bar{\mathbf{y}}. \tag{12}$$

such that $\mathbf{x}_i = \bar{\mathbf{x}}_j$.

As a result of this corollary, instead of using a $O(n^3)$-time and $O(n^2)$-space algorithm for computing the solution $\mathbf{f}^*$ as in Eq. (11), we can obtain an exact solution by computing $\bar{\mathbf{f}}^*$ in Eq. (12) which involves matrices of size $k \times k$; hence, $O(k^3)$-time and $O(k^2)$-space complexities. Therefore, if k is small, we have a much more efficient algorithm.

In practice, however, it is rare to obtain two identical fingerprints even at the same location due to noise and so if we use the exact multi-set representation for the fingerprint sequence the value of k can be as large as n. In what follows, we present an approximate algorithm that uses a fixed-size buffer of up to k representative fingerprints where k is set to a predefined constant.

5.2 Representative Buffer

Hereafter, we assume that k is a constant. At any time n we maintain a buffer

$$B_n = \{(\tilde{\mathbf{x}}_i, \tilde{m}_i, \tilde{h}_i, \tilde{y}_i)\}_i$$

of up to k representative fingerprints $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_k \in \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, where there is a count $\tilde{m}_i$ for $\tilde{\mathbf{x}}_i$ for the number of fingerprints represented by $\mathbf{x}_i$, and another count $\tilde{h}_i$ for the number of labeled fingerprints therein. Each fingerprint is assigned to 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".

Natural candidates to be the representative fingerprints in the buffer are the k-centers (as in the well-known metric k-center clustering problem),

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

The k-center representatives are 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).

For a fingerprint stream, its k-centers can be computed, and updated incrementally upon receipt of each new fingerprint, by an incremental k-center clustering algorithm. In particular, we use Charikar et al.'s Doubling Algorithm [7]. The Doubling Algorithm guarantees that the distance between a fingerprint and its assigned representative is always less than the distance between any pair of centers. When applying the Doubling Algorithm, the details being listed in Algorithm 1, 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; see lines 6-11, and (2) in the re-clustering process when the number of centers exceeds k, labeled fingerprints are selected first to serve as center whenever needed; see line 23. More presence of labeled fingerprints in the representative set should result in more information useful for the location prediction.

It is possible that the buffer consists of $k' < k$ centers, for example, during the initial phase of the fingerprint stream or as a result of the re-clustering using the Doubling Algorithm; in this case, location estimation is computed using these k' centers. For simplicity of presentation, we assume below that the buffer has k representative fingerprints.

5.3 Location Estimation

Similar to how we define the matrices $\bar{\mathbf{H}}$, $\bar{\mathbf{L}}$, $\bar{\mathbf{B}}$ and $\bar{\mathbf{K}}$ for the exact multi-set, we have the corresponding matrices $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{K}}$ for the representative fingerprints in B_n. When each new fingerprint is available, the buffer needs be updated and, accordingly, so do these matrices. Matrices $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, and $\tilde{\mathbf{K}}$ can be computed easily after the buffer is updated, as follows:

$$\tilde{\mathbf{H}} = diag(\tilde{h}_1, \tilde{h}_2, .., \tilde{h}_k)$$

$$\tilde{\mathbf{L}} = \left[\tilde{l}_{ij} = \begin{cases} -\tilde{m}_i \tilde{m}_j w(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) & \text{if } i \neq j \\ \sum_{p=1}^{k} \tilde{m}_i \tilde{m}_p w(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_p) & \text{otherwise} \end{cases} \right]_{k \times k}$$

Algorithm 1: Incremental Update of Representative Buffer

/* run at time n after obtaining fingerprint $\mathbf{x}_n$ */
Input: fingerprint $\mathbf{x}_n$, representative buffer $B_{n-1} = \{(\tilde{\mathbf{x}}_i, \tilde{m}_i, \tilde{y}_i)\}_i$
Output: new representative buffer B_n
if *(n equals 1)* **then**
  $B_1 = \{(\mathbf{x}_1, 1, h(\mathbf{x}_1), y(\mathbf{x}_1)\}$;
  Initialize R to a small value, e.g., 0.0001;
  return;
end
Find the labeled center $\tilde{\mathbf{x}}_i$ in B_{n-1} s.t. $\|\mathbf{x}_n - \tilde{\mathbf{x}}_i\| < R$ and $\|\mathbf{x}_n - \tilde{\mathbf{x}}_i\|$ is minimum;
if *(that $\tilde{\mathbf{x}}_i$ is found)* **then**
  /* Assign $\mathbf{x}_n$ to center $\tilde{\mathbf{x}}_i$ */
  $\tilde{m}_i$ ++;
  $B_n = B_{n-1}$;
  return;
end
Find the unlabeled center $\tilde{\mathbf{x}}_i$ in B_{n-1} s.t. $\|\mathbf{x}_n - \tilde{\mathbf{x}}_i\| < R$ and $\|\mathbf{x}_n - \tilde{\mathbf{x}}_i\|$ is minimum;
if *(that $\tilde{\mathbf{x}}_i$ is found)* **then**
  /* Assign $\mathbf{x}_n$ to center $\tilde{\mathbf{x}}_i$ */
  $\tilde{m}_i$ ++;
  $B_n = B_{n-1}$;
  return;
end
/* If we get here, $\mathbf{x}_n$ is too far, at least R away, from any center; create a new center */
$B_n = B_{n-1} \cup \{(\mathbf{x}_n, 1, h(\mathbf{x}_n), y(\mathbf{x}_n))\}$;
while *($|B_n| > k$)* **do**
  /* Need to re-cluster */
  $R = 2 \times R$;
  $B_{temp} = \varnothing$;
  repeat
    Choose one center $\tilde{\mathbf{x}}_i$ in B_n, preferably labeled;
    $B_n = B_n \setminus \{(\tilde{\mathbf{x}}_i, \tilde{m}_i, \tilde{h}_i, \tilde{y}_i)\}$;
    for *(each center $\tilde{\mathbf{x}}_l$ in B_n s.t. $\|\tilde{\mathbf{x}}_l - \tilde{\mathbf{x}}_i\| < R$)* **do**
      /* Collapse center $\tilde{\mathbf{x}}_l$ into center $\tilde{\mathbf{x}}_i$ */
      $\tilde{m}_i = \tilde{m}_i + \tilde{m}_l$;
      $B_n = B_n \setminus \{(\tilde{\mathbf{x}}_l, \tilde{m}_l, \tilde{h}_l, \tilde{y}_l)\}$;
    end
    $B_{temp} = B_{temp} \cup \{(\tilde{\mathbf{x}}_i, \tilde{m}_i, \tilde{h}_i, \tilde{y}_i)\}$;
  until *($B_n = \varnothing$)*;
  $B_n = B_{temp}$;
  return;
end

$$\tilde{\mathbf{K}} = \left[\tilde{k}_{ij} = \exp\left(-\frac{|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j|^2}{2\gamma^2} \right) \right]_{k \times k}$$

Let "$\mapsto$" denote the assignment of a fingerprint to a representative. Matrix $\tilde{\mathbf{B}}$ is the following matrix

$$\tilde{\mathbf{B}} = \left[\tilde{b}_{ij} = \sum_{p|\mathbf{x_p}\mapsto\tilde{\mathbf{x}}_i} \sum_{q|\mathbf{x_q}\mapsto\tilde{\mathbf{x}}_j} b_{pq} \right]_{k\times k}. \tag{13}$$

The entries $\{\tilde{b}_{ij}\}_{k\times k}$ are incrementally updated during the same time as the buffer is being updated. Initially, at time zero, $\tilde{b}_{ij}$ is set to 0 for every $i, j \leq k$. Suppose that the current buffer is $\{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_k\}$ when we receive a new fingerprint $\mathbf{x}_n$. Because we use the Doubling Algorithm for updating the k-centers, 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, as a result of using the Doubling Algorithm, suppose that $\mathbf{x}_n$ is assigned to some representative $\tilde{\mathbf{x}}_i$ (see line 7 or line 13 in Algorithm 1). Because of this new assignment, only the values of $\tilde{b}_{ij}$ and $\tilde{b}_{ji}$ for $j = 1, 2, ..., k$ need to be updated such that

$$\begin{aligned} \tilde{b}_{ij} += & \sum_{q|\mathbf{x_q}\mapsto\tilde{\mathbf{x}}_j} b_{nq} \\ \tilde{b}_{ji} += & \sum_{q|\mathbf{x_q}\mapsto\tilde{\mathbf{x}}_j} b_{nq}. \end{aligned}$$

Recall that in matrix $\mathbf{B}$ entry b_{pq} is zero everywhere except when $|p-q| \leq 2$. Therefore, $\tilde{b}_{ij}$ (and $\tilde{b}_{ji}$) can be efficiently computed as follows:

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

For this computation to be possible, we need to keep track of which fingerprints in the buffer represent the two latest fingerprints, $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n-2}$.

It is also important to note that at each new step n, the entry $b_{n-1,n-1}$ changes value from 1 (value at time $n-1$) to 5 (value at time n). Similarly, $b_{n-2,n-2}$ changes value from 5 to 6, and $b_{n-1,n-2}$ and $b_{n-2,n-1}$ both from -2 to -4. The values of the other entries of matrix B are intact. Consequently, supposing that $\mathbf{x}_{n-1} \mapsto \tilde{\mathbf{x}}_{j_1}$ and $\mathbf{x}_{n-2} \mapsto \tilde{\mathbf{x}}_{j_2}$, we need to make the following updates:

$$\begin{aligned} \tilde{b}_{j_1 j_1} += & 4 \\ \tilde{b}_{j_2 j_2} += & 1 \\ \tilde{b}_{j_1 j_2} -= & 2 \\ \tilde{b}_{j_2 j_1} -= & 2. \end{aligned}$$

The time complexity 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 when a representative $\tilde{\mathbf{x}}_l$ is collapsed into another

Algorithm 2: Online Algorithm

/* run at time n after fingerprint $\mathbf{x}_n$ is observed */
Input: fingerprint $\mathbf{x}_n$
Output: location of $\mathbf{x}_n$
1 Update buffer $B_n = \{(\tilde{\mathbf{x}}_i, \tilde{m}_i, \tilde{h}_i, \tilde{y}_i)\}_{i=1}^{k}$ according to Algorithm 1 and at the same time update matrix $\tilde{\mathbf{B}}$;
2 **if** *(* $\mathbf{x}_n$ *is labeled)* **then**
3 Location of $\mathbf{x}_n$ is the given y_n;
4 **return**;
5 **end**
6 Compute $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, and $\tilde{\mathbf{K}}$;
7 Let $\tilde{\mathbf{y}} = [\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_k]^\intercal$ where $\tilde{y}_i$ is the given location of representative fingerprint $\tilde{\mathbf{x}}_i$ if it is labeled and set to zero otherwise; $\bar{\mathbf{I}}$ be the $n \times n$ identity matrix;
8 Compute $\tilde{\mathbf{f}} = \left(\lambda_K \bar{\mathbf{I}} + \tilde{\mathbf{K}}(\tilde{\mathbf{H}} + \lambda_S \tilde{\mathbf{L}} + \lambda_T \tilde{\mathbf{B}})\right)^{-1} \tilde{\mathbf{K}} \tilde{\mathbf{H}} \tilde{\mathbf{y}}$;
9 Location of $\mathbf{x}_n$ is the element in $\tilde{\mathbf{f}}$ that corresponds to the representative fingerprint of $\mathbf{x}_n$;
10 **return**;

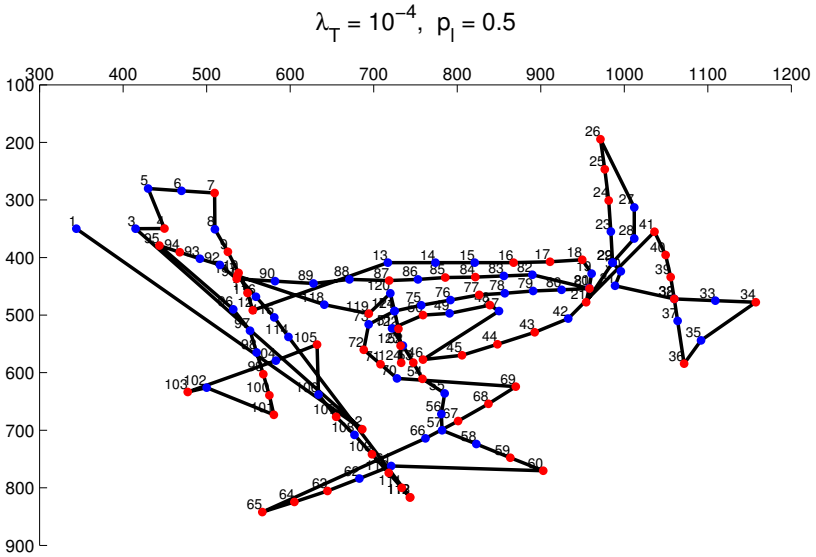
representative $\tilde{\mathbf{x}}_i$ (see line 25 in Algorithm 1), we make the following update:

$$\begin{aligned} \forall\, j : 1 \le j \le k \wedge j \neq l : & \\ \tilde{b}_{ij} &+= \tilde{b}_{jl} \\ \tilde{b}_{ji} &+= \tilde{b}_{jl} \\ \forall\, j : 1 \le j \le k : & \\ \tilde{b}_{lj} &= 0 \\ \tilde{b}_{jl} &= 0. \end{aligned}$$

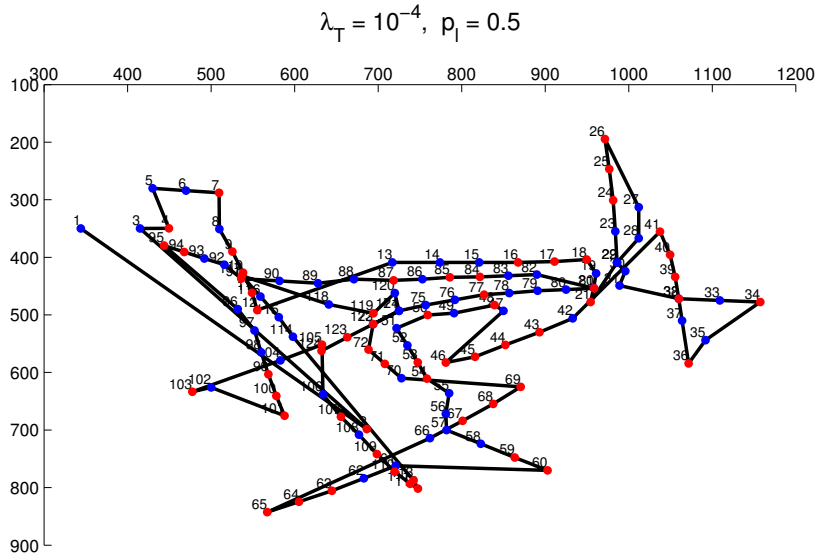
The time complexity for updating $\tilde{\mathbf{B}}$ in the re-clustering step, if it occurs, is also $O(k)$.

After matrices $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{K}}$ have been updated the location estimation is simply an application of Eq. (12) in Corollary 1 where we use $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{K}}$ instead of $\bar{\mathbf{H}}$, $\bar{\mathbf{L}}$, $\bar{\mathbf{B}}$, and $\bar{\mathbf{K}}$. This is summarized in Algorithm 2. The total time complexity for estimating the location of each fingerprint is dominated by the time to compute the inverse matrix; hence, $O(k^3)$.

The promise of this online algorithm has been substantiated by Tran and Zhang in [27]. Figure 11 shows an example where given a sequence of 124 fingerprints, 50% of which labeled, the estimated trajectory using a buffer storing only 20% of the sequence length can closely approximate the trajectory computed using an infinite buffer. For this result, only temporal regularization is applied and the ground-truth trajectory is shown in Figure 12; the parameters are temporal regularization coefficient $\lambda_T = 10^{-4}$, kernel regularization coefficient $\lambda_K = 10^{-6}$, and Gaussian parameters $\gamma = 1$ and $\sigma = 0.1$ for the kernel function and weight function, respectively.



(a) Buffer size: 20% sequence length



(b) Buffer size: infinite

Fig. 11 Example of the estimated trajectory for a fingerprint sequence with 50% of the fingerprints labeled. Red-colored points are location estimates for the unlabeled fingerprints and blue-colored points are the ground-truth locations of the labeled fingerprints. The numbers represent the IDs of the fingerprints sorted in time of observation. The ground-truth trajectory is shown in Figure 12. ([27])

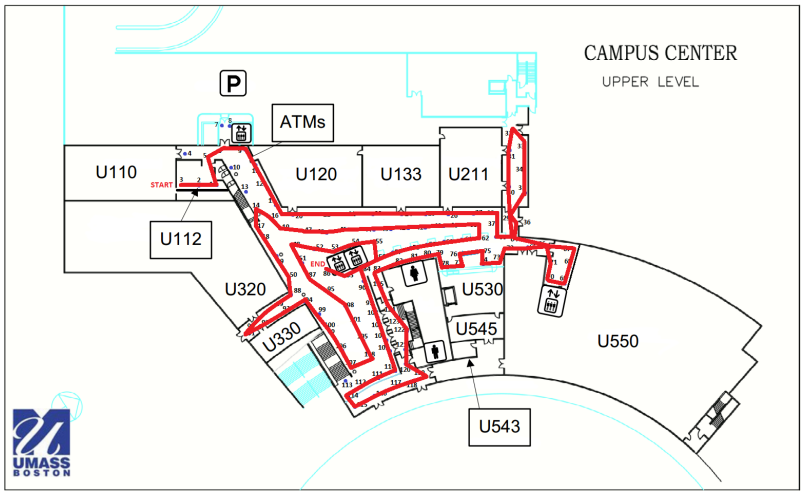


Fig. 12 A test trajectory visiting 124 locations on the Upper Level (185m $\times$ 113m) in the Campus Center at UMass Boston.

In terms of computation time, Figure 13 provides an illustration of the time advantage of the online algorithm for this trajectory. The time to locate each individual fingerprint 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, which does not increase even when the 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.

6 Conclusions

Fingerprint-based localization and tracking can be made more effective by enriching the set of training fingerprints and by utilizing real-time fingerprints obtainable on the spot. To realize this potential, a regularization approach has been presented, which, by incorporating spatial and temporal characteristics of the fingerprint space into the regularization framework, can offer promising estimation accuracy. Also discussed is an algorithm with constant-bounded computational complexities which can be used to track the location of a moving device in real time.

Potential applications of the presented regularization frameworks 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

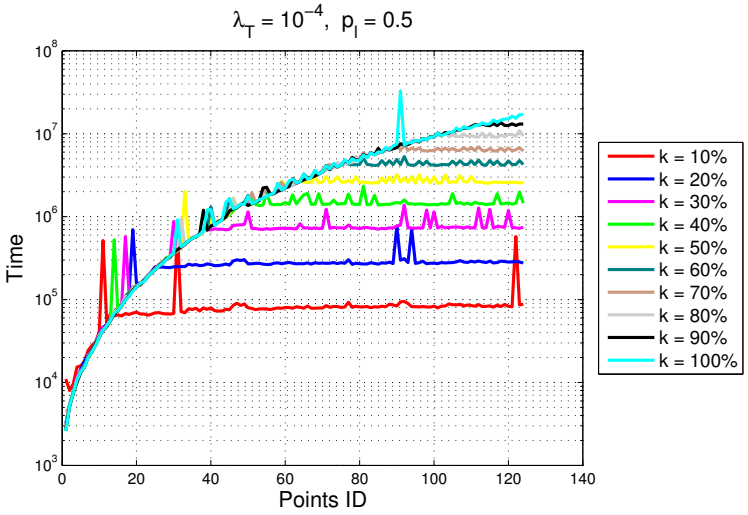


Fig. 13 Localization time (in log scale) to locate each individual fingerprint. Buffer size k is expressed as a percentage of the length of the fingerprint sequence. ([27])

the phone can read automatically when passing nearby; this labeled location information can be used to infer location at any other place. In tracking of autonomous underwater vehicles (AUVs) deployed in underwater environments, the AUV while submerged is tracked using a built-in inertial navigation system that has to dead reckon with GPS each time the AUV surfaces; it is desirable to improve the localization accuracy of the AUV between these GPS fixes. It is noted that in all these applications, the training data comes sequentially in real time.

Although theoretically interesting, there exist challenges when implementing a regularization framework. For example, how to find the best regularization coefficients, or in the case of online tracking, what should be the best fingerprint buffer size? Assuming a sparse fingerprint matrix, we conjecture that the best fingerprint buffer size should be logarithmic to the length of the fingerprint sequence. Another challenge is to determine the best label rate, i.e., the amount of labeled fingerprints relatively to the amount of unlabeled, as too high or too low a rate may degrade the effectiveness of the regularization. Also, in practice, there are other constraints regarding the mobility of the moving device that should be incorporated into the framework for better accuracy. Consequently, there is much more room for future research. In the mean time, the presented techniques can be used to provide benchmark for evaluation of location fingerprinting.

Acknowledgements This work was supported by the NSF award CNS-1116430. Any opinions, findings and conclusions or recommendations expressed in this material are ours and do not necessarily reflect those of the NSF.

References

1. Bahl, P., Padmanabhan, V.N.: Radar: An in-building rf-based user location and tracking system. In: INFOCOM, pp. 775–784 (2000)
2. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. **7**, 2399–2434 (2006).
3. Boukerche, A., Oliveira, H.A.B.F., Nakamura, E.F., Loureiro, A.A.F.: Vehicular ad hoc networks: A new challenge for localization-based systems. Comput. Commun. **31**(12), 2838–2849 (2008). DOI 10.1016/j.comcom.2007.12.004.
4. Bresson, X., Zhang, R.: Tv-svm: Total variation support vector machine for semi-supervised data classification. CoRR **abs/1210.0699** (2012)
5. Brunato, M., Battiti, R.: Statistical learning theory for location fingerprinting in wireless lans. Comput. Netw. **47**(6), 825–845 (2005). DOI 10.1016/j.comnet.2004.09.004.
6. Chapelle, O., Schlkopf, B., Zien, A.: Semi-Supervised Learning, 1st edn. The MIT Press (2010)
7. Charikar, M., Chekuri, C., Feder, T., Motwani, R.: Incremental clustering and dynamic information retrieval. In: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, STOC '97, pp. 626–635. ACM, New York, NY, USA (1997). DOI 10.1145/258533.258657
8. Chen, Y., Lymberopoulos, D., Liu, J., Priyantha, B.: Fm-based indoor localization. In: Proceedings of the 10th international conference on Mobile systems, applications, and services, MobiSys '12, pp. 169–182. ACM, New York, NY, USA (2012). DOI 10.1145/2307636.2307653.
9. Chung, J., Donahoe, M., Schmandt, C., Kim, I.J., Razavai, P., Wiseman, M.: Indoor location sensing using geo-magnetism. In: Proceedings of the 9th international conference on Mobile systems, applications, and services, MobiSys '11, pp. 141–154. ACM, New York, NY, USA (2011). DOI 10.1145/1999995.2000010.
10. Elmoataz, A., Lezoray, O., Bougleux, S.: Nonlocal discrete regularization on weighted graphs: A framework for image and manifold processing. Trans. Img. Proc. **17**(7), 1047–1060 (2008). DOI 10.1109/TIP.2008.924284.
11. Figuera, C., Rojo-Álvarez, J.L., Wilby, M., Mora-Jiménez, I., Caamaño, A.J.: Advanced support vector machines for 802.11 indoor location. Signal Process. **92**(9), 2126–2136 (2012). DOI 10.1016/j.sigpro.2012.01.026.
12. Hu, P., Li, L., Peng, C., Shen, G., Zhao, F.: Pharos: Enable physical analytics through visible light based indoor localization. In: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, HotNets-XII, pp. 5:1–5:7. ACM, New York, NY, USA (2013). DOI 10.1145/2535771.2535790.
13. Laoudias, C., Eliades, D.G., Kemppi, P., Panayiotou, C.G., Polycarpou, M.M.: Indoor localization using neural networks with location fingerprints. In: Proceedings of the 19th International Conference on Artificial Neural Networks: Part II, ICANN '09, pp. 954–963. Springer-Verlag, Berlin, Heidelberg (2009)
14. Laufer-Goldshtein, B., Talmon, R., Gannot, S.: Manifold-based bayesian inference for semi-supervised source localization. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6335–6339 (2016). DOI 10.1109/ICASSP.2016.7472896
15. Lin, T., Xue, H., Wang, L., Zha, H.: Total variation and euler's elastica for supervised learning. In: ICML (2012)
16. Lorincz, K., Welsh, M.: Motetrack: a robust, decentralized approach to rf-based location tracking. Personal Ubiquitous Comput. **11**(6), 489–503 (2007). DOI 10.1007/s00779-006-0095-2.
17. Pan, J.J., Yang, Q.: Co-localization from labeled and unlabeled data using graph laplacian. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, pp. 2166–2171. Hyderabad, India (2007)
18. Pan, J.J., Yang, Q., Chang, H., Yeung, D.Y.: A manifold regularization approach to calibration reduction for sensor-network based tracking. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence, pp. 988–993. Boston, United States (2006)

19. Pulkkinen, T., Roos, T., Myllymäki, P.: Semi-supervised learning for wlan positioning. In: Proceedings of the 21th international conference on Artificial neural networks - Volume Part I, ICANN'11, pp. 355–362. Springer-Verlag, Berlin, Heidelberg (2011).
20. Rallapalli, S., Qiu, L., Zhang, Y., Chen, Y.C.: 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, MobiCom '10, pp. 161–172. ACM, New York, NY, USA (2010). DOI 10.1145/1859995.1860015.
21. Roos, T., Myllymäki, P., Tirri, H., Misikangas, P., Sievänen, J.: A probabilistic approach to WLAN user location estimation. International Journal of Wireless Information Networks **9**(3), 155–164 (2002). DOI 10.1023/A:1016003126882.
22. Schölkopf, B., Herbrich, R., Smola, A.J.: A generalized representer theorem. In: Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory, COLT '01/EuroCOLT '01, pp. 416–426. Springer-Verlag, London, UK, UK (2001).
23. Tarzia, S.P., Dinda, P.A., Dick, R.P., Memik, G.: Indoor localization without infrastructure using the acoustic background spectrum. In: Proceedings of the 9th international conference on Mobile systems, applications, and services, MobiSys '11, pp. 155–168. ACM, New York, NY, USA (2011). DOI 10.1145/1999995.2000011.
24. Tran, D.A., Nguyen, T.: Localization in wireless sensor networks based on support vector machines. IEEE Transactions on Parallel and Distributed Systems **19**(7), 981–994 (2008)
25. Tran, D.A., Truong, P.: Total variation regularization for training of indoor location fingerprints. In: Proceedings of ACM MOBICOM Workshop on Mission-Oriented Wireless Sensor Networking (ACM MiseNet 2013). Miami (2013)
26. Tran, D.A., Zhang, T.: Fingerprint-based location tracking with hodrick-prescott filtering. In: Proceedings of the 7th IFIP Wireless and Mobile Networking Conference (WMNC 2014). Algarve, Portugal (2014).
27. Tran, D.A., Zhang, T.: An online algorithm for fingerprint-based location tracking. In: Proceedings of the 11th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2014). Philadelphia, PA (2014).
28. Tran, D.A., Zhang, T., Gong, S.: A regularization framework for fingerprint-based reconstruction of mobile trajectories. Int. J. Parallel Emerg. Distrib. Syst. **31**(3), 268–279 (2016). DOI 10.1080/17445760.2015.1085036.
29. Varshavsky, A., de Lara, E., Hightower, J., LaMarca, A., Otsason, V.: Gsm indoor localization. Pervasive Mob. Comput. **3**(6), 698–720 (2007). DOI 10.1016/j.pmcj.2007.07.004.
30. Whitehouse, K., Karlof, C., Culler, D.: A practical evaluation of radio signal strength for ranging-based localization. SIGMOBILE Mob. Comput. Commun. Rev. **11**(1), 41–52 (2007). DOI 10.1145/1234822.1234829.
31. Wu, C., Yang, Z., Liu, Y., Xi, W.: Will: Wireless indoor localization without site survey. IEEE Transactions on Parallel and Distributed Systems **24**(4), 839–848 (2013). DOI http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.179
32. Wu, C.L., Fu, L.C., Lian, F.L.: WLAN location determination in e-home via support vector classification. In: Networking, Sensing and Control, 2004 IEEE International Conference on, vol. 2, pp. 1026–1031 Vol.2 (2004).
33. Yang, Z., Wu, C., Liu, Y.: Locating in fingerprint space: wireless indoor localization with little human intervention. In: Proceedings of the 18th annual international conference on Mobile computing and networking, Mobicom '12, pp. 269–280. ACM, New York, NY, USA (2012). DOI 10.1145/2348543.2348578.
34. jia Zhu, Y., liang Deng, Z., Ji, H.: Indoor localization via l1-graph regularized semi-supervised manifold learning. The Journal of China Universities of Posts and Telecommunications **19**(5), 39 – 91 (2012). DOI 10.1016/S1005-8885(11)60298-7.