Leto: Crowdsourced Radio Map Construction With Learned Topology and a Few Landmarks

Yiwen Wang, *Graduate Student Member, IEEE*, Albert Kai-Sun Wong, *Member, IEEE*, S.-H. Gary Chan, *Senior Member, IEEE*, and Wai Ho Mow, *Senior Member, IEEE*

***Abstract*—Existing crowdsourced indoor positioning systems (CIPSs) usually require prior knowledge about the site and a tedious calibration process. Moreover, they may require a large number of landmarks while ignoring the topology information that may be contained in the crowdsourced data. In this paper, we present Leto, a system that uses learned topology information from combined user traces to construct a radio map. Leto relies on crowdsourced WiFi and accelerometer signals only without requiring any prior knowledge about the site. Our key idea is that learned topology information can reduce the required number of landmarks, while available landmarks can transform the topology into a map. We propose a novel framework that efficiently learns the map topology by a hybrid multidimensional scaling (HMDS) algorithm and accurately rectifies the map using only a few anchors by an adaptive force-directed (AFD) algorithm. We also provide a theoretical convergence analysis of the HMDS algorithm. Experimental results on real-world datasets show that Leto can capture useful topology information and achieve significant improvements in radio map construction compared to existing systems.**

***Index Terms*—Map rectification, radio map construction, topology learning.**

I. Introduction

INDOOR Positioning System, or IPS, plays a fundamental role in emerging indoor mobile applications such as indoor navigation, geo-fencing, contact tracing, smart buildings, and virtual and augmented reality (VR/AR). While infrastructure-based IPSs, which require installation of additional equipment such as Bluetooth beacons, UWB base stations and cameras, incur extra deployment effort and cost, WiFi fingerprinting-based IPSs [1], [2], [3] exploit the ubiquitous presence of WiFi access points (APs). WiFi fingerprinting IPSs have demonstrated good localization capabilities in many complicated environments, but they require manual surveys in an offline phase to create a *radio map* that contains the coordinates and WiFi measurements at many locations. Moreover, they require an update of the radio map when there are AP and environmental changes.

CIPSs [4], [5], [6], [7], [8], [9], [10], [11], [12], or crowdsourced IPSs, aim to eliminate manual surveys and dynamically adapt to AP changes. SLAM (Simultaneous Localization and Mapping) [13], [14], [15], [16], [17], [18], [19], first developed within the robotic community, is the process by which a mobile robot uses odometry, IMU (inertia measurement unit), range sensing, and computer vision to build a map of the environment while using this map to locate itself. Inspired by SLAM, researchers in CIPS proposed the use of crowd PDR (Pedestrian Dead Reckoning) for radio map construction, where PDR is the process of calculating a user's; position based on her previous position, estimated heading direction, and displacement [5], [6]. It is well known that PDR trajectories calculated from IMU measurements can suffer from drift errors over the long term. To create accurate radio maps, CIPSs may need many landmarks, which are identifiable signatures on one or more sensing dimensions, to recalibrate user trajectories. To reduce the need of landmarks, many CIPSs [4], [7], [8], [9], [10], [11], [12] map the user trajectories individually or jointly to a floor plan, which may not always be available in practice.

More recently, GraphIPS [20], a graph-based SLAM system, was proposed for constructing a radio map without a floor plan and with reduced dependency on IMU. GraphIPS exploits information contained in WiFi measurements to infer pairwise distances between locations. With these spatial constraints, it then uses multidimensional scaling (MDS) to determine the coordinates of the locations. However, GraphIPS assumes that the AP positions are known and AoA measurements are available. Furthermore, it is not clear whether a reasonable radio map can be generated if the pairwise distance matrix is highly incomplete.

Our system, Leto, for **le**arned **to**pology, assumes the scenario shown in Fig. 1. When users move around the environment, apps running on their mobile devices collect and upload *user traces*, which are sequences of time-stamped WiFi RSSI and accelerometer measurements. Following the idea of graph-based SLAM [13], we use a graph to represent the radio map. Each *node* in the graph corresponds to one measurement associated with a location. Each *edge* has a weight that corresponds to the physical distance between the two nodes. Leto makes use of three types of distances: *consecutive intra-trace*

Manuscript received 8 October 2022; revised 17 March 2023; accepted 3 April 2023. Date of publication 11 April 2023; date of current version 6 March 2024. This work was supported in part by Hong Kong General Research Fund under Grant 16200120. Recommended for acceptance by Y. Zhu. *(Corresponding author: Yiwen Wang.)*

Yiwen Wang, Albert Kai-Sun Wong, and Wai Ho Mow are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: ywangda@ust.hk; eealbert@ust.hk; eewhmow@ust.hk).

S.-H. Gary Chan is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: gchan@cse.ust.hk).

Digital Object Identifier 10.1109/TMC.2023.3266198

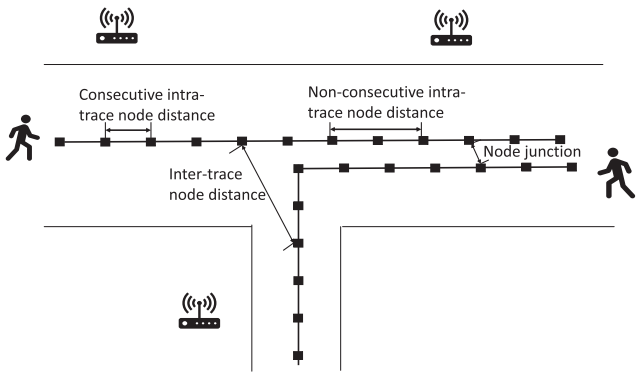


Fig. 1. Leto user traces and distance model.

node distance, non-consecutive intra-trace node distance, and *inter-trace node distance*. These distances, or spatial constraints, may contain different levels of uncertainty. We build the graph, or a topological map, by finding a node configuration that best satisfies the constraints. Then, we rectify the topological map by using a few landmarks. Our radio map construction framework is shown in Fig. 2 and will be described in detail in Section III.

We summarize our major contributions as follows:

- Leto fuses crowdsourced WiFi signals, accelerometer signals, and a few landmarks to build a radio map. Compared to CIPSs using full IMU signals, Leto is less dependent on mobile devices being held steadily to produce quality IMU signals. Furthermore, Leto does not require a floor plan, making it more widely applicable.
- We introduce a number of new features for distance estimation in graph-based CIPSs. These features include stride parameters for individual traces, a two-phase scheme for estimating pairwise distances among WiFi measurements by first applying short- and long-distance classification, and the incorporation of uncertainty measures in our stress minimization formulation.
- We introduce a low-complexity modified Smith-Waterman (mSW) algorithm to detect junctions between two WiFi sequences. By applying a sliding window technique, mSW achieves $O(mn)$ time complexity and $O(\max\{m, n\})$ space complexity, where m, n are lengths of the two sequences.
- We introduce a low-complexity HMDS algorithm for stress minimization, which is a challenging non-convex optimization problem. By stress minimization, the topological map and individual stride parameters can be obtained simultaneously. We also provide the convergence and computation complexity analysis of the proposed algorithm.
- We introduce an AFD algorithm to rectify the topological map using a few landmarks. The AFD algorithm dynamically selects neighboring nodes to escape from local minima and adaptively adjusts the step size to speed up convergence. We also provide a complexity analysis of the AFD algorithm.
- Experiments are conducted to compare Leto against five state-of-the-art CIPSs and to verify the effectiveness and convergences of the proposed HMDS and AFD algorithms. It is shown that Leto achieves significant performance improvement in terms of map and localization accuracy and also effectively reduces the assumptions by other CIPSs.

II. Related Work

Crowdsourced IPSs (CIPSs) based on PDR have attracted significant attention in recent years. To mitigate the effect of IMU drift, several CIPSs, including Unloc [5], Walkie-Markie [6] and WiFi-RITA [21], propose the use of detected landmarks to correct the estimated PDR trajectories. But these systems assume that mobile devices are held in a steady fashion. Also, accuracy of the radio map created highly depends on the richness of detected landmarks. This prompted CIPSs including Zee, LiFs, and others to incorporate extra information from floor plans [4], [8], [9], [10], [11], [12], [22].

A floor plan can be represented as a probabilistic model [4], [9]. In zee [4], an augmented particle filter algorithm is used to combine sensor information with constraints imposed by the floor plan in estimating user locations. In [9], a hidden Markov model (HMM) is used to describe the user traces constrained by the floor plan. Nevertheless, the complexity of the learning probabilistic models is very high. This complexity limits the use of probabilistic models to small indoor areas.

To enhance scalability, many CIPSs use a graph-based model [7], [8], [10], [11], [12], [22] to represent a floor plan. In [22], a logical floor plan is first constructed by exploiting the relationships among different rooms. Then, the logical floor plan is mapped to a physical floor plan by graph matching techniques. In [7], a simulated radio map is generated by a simulator that requires details of the indoor environment, including the floor plan, wall materials, and furniture. Then, the simulated radio map is transferred to a limited number of calibration fingerprints by manifold alignment techniques. In LiFs [8], a stress-free floor plan is created in a high-dimensional space such that the walking distances between every pair of locations are preserved. Similarly, a fingerprint space is created such that the mutual distances between fingerprints are preserved. Finally, the fingerprint space is mapped to the stress-free floor plan by analyzing the spatial similarity. In [10], a topological radio map is created by analyzing the spatial correlation of massive user traces. The radio map is then mapped to the floor map by graph matching techniques. [10] further studies the issue of symmetries and the number and positions of markers required to associate a topological radio map with a physical floor plan unambiguously. To reduce the signal bias and labeling error, UbiFin [11] fuses crowdsourced RF, magnetic field, and motion signals. Traces are partitioned into segments and mapped to the physical floor plan by dynamic programming. In CRCLoc [12], all possible paths in a floor plan are extracted by image processing techniques. Then, a shape context algorithm is adopted to map the estimated user trajectories to the candidate paths.

Although floor plans can provide structural information, they may not be available or up-to-date in practice. GraphIPS [20] removes the requirement of a floor plan and exploits the spatial relationships between users and APs. GraphIPS fuses WiFi and IMU signals into a graph-based formulation and adopts MDS to estimate user location. But its assumptions that AP locations

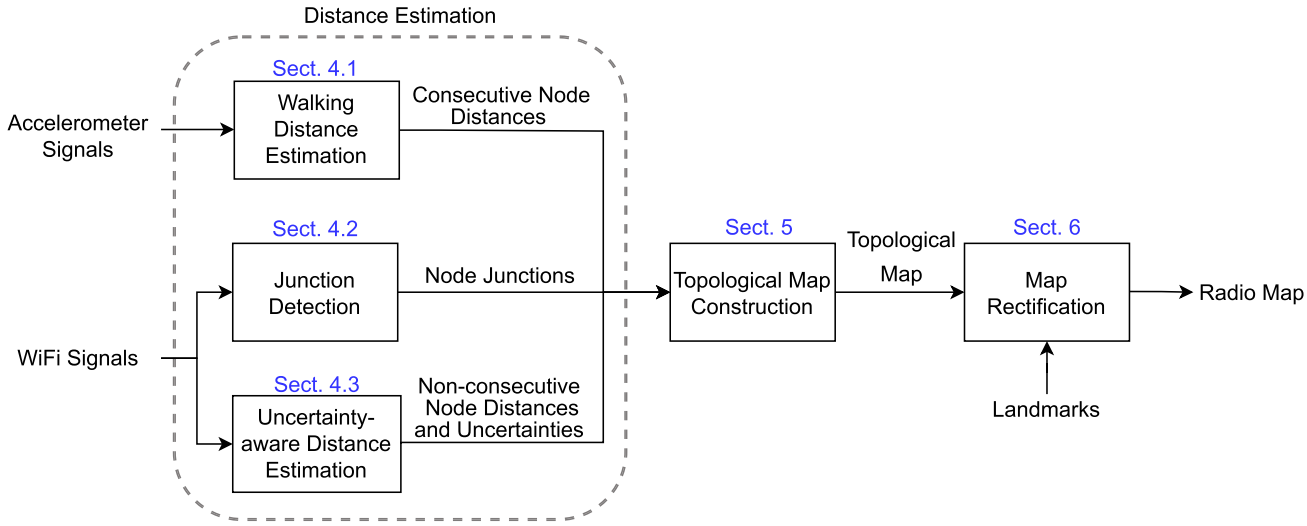


Fig. 2. A diagram illustrating the full pipeline of radio map construction for Leto.

are known and WiFi AoA data is available may limit its use in practice.

We will compare Leto against five previously proposed crowdsourced systems mentioned above: Unloc, Zee, LiFs, WiFi-RITA, and GraphIPS [4], [5], [8], [20], [21].

III. System Overview

Leto constructs a radio map using WiFi signals, accelerometer signals, and a few landmarks for low cost, wide availability, and small deployment effort. In this work, we focus on radio map construction on a single floor. However, it is easy to extend our framework for a multi-floor setting by incorporating floor identification techniques [23], [24] into our framework.

The radio map construction framework for Leto is shown in Fig. 2 and will be described in Section IV, V, and VI. Our distance estimation framework consists of three parts:

1. Use of the accelerometer signals to estimate all the consecutive node distances (Section 4.1). These distance estimates are first parametrized by two unknown stride parameters specific to each trace.
2. Use of our modified Smith-Waterman algorithm (Section 4.2) for junction detection - to identify node junctions, or pairs of WiFi measurements in different traces that are likely to be taken at identical locations.
3. Use of a collection of neural networks, which we call UDENet (Section 4.3), to estimate as many non-consecutive node distances as we can. UDENet first classifies a distance to be a short or a long distance. Then, a separate neural network is applied to estimate the distance in each class. While estimating each distance, UDENet also estimates the associated uncertainty, and this uncertainty will be used as a weight in our optimization problem for topological map construction.

Section V describes our topological map construction. Here, the parametrized and absolute distance estimates from Section IV are used as inputs to the efficient HMDS algorithm we developed. The output is the complete set of relative 2D coordinates of all the nodes and the stride parameter values for all traces. We call this output a topological map because while the relative coordinates and connectivity (as provided by the traces) are determined, the absolute coordinates are not yet known.

Then, Section VI describes the final step, map rectification. Here, we use the relative coordinates plus the absolute coordinates of a few landmarks as input to our AFD algorithm to re-calibrate the relative coordinates as well as to map the relative coordinates to the absolute coordinates. We will show that this rectification process can allow us to construct a final map that can be highly consistent with the unknown underlying floor plan.

In Section VII, we provide experimental results comparing the Leto against the five existing systems in two types of environments – a campus with many corridors and classrooms, and a shopping mall with shops and open spaces. In Section VIII, we provide the conclusion.

Table I shows the notations used throughout this paper:

IV. Distance Estimation

This section presents distance estimation using accelerometer and WiFi signals. Using accelerometer signals, we detect steps and estimate the walking distance between consecutive nodes in each trace. Using WiFi signals, we detect junctions among traces and estimate pairwise distances between non-consecutive nodes.

A. Walking Distance Estimation

Walking distance estimation allows us to establish the constraints among nodes within the same trace. To estimate walking distances, we first detect and count steps. Many step detection algorithms, e.g., peak detection, zero-crossing, detection, and spectrum analysis, have been proposed in the literature. In Leto,we employ the step detection algorithm proposed in [25].

Since the position and orientation of the mobile device may change when a user is collecting data, we use the magnitude

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Description
$\mathbf{S}$	Substitution matrix for junction detection
$\mathbf{H}$	Scoring matrix for junction detection
$\mathbf{fp}$	WiFi fingerprint vector
$\mathcal{A}$	Set of APs
M_{trace}	Number of traces
K_{AP}	Number of APs
$\mathbf{X}$	2D coordinates of all nodes
$\mathcal{I}$	Set of all pairs of non-consecutive nodes
$\mathcal{J}_m$	Set of all pairs of consecutive nodes for trace m
$d_{i,j}$, $\hat{d}_{i,j}$	True and estimated distances between non-consecutive nodes i and j
$d_{i,i+1}$, $\hat{d}_{i,i+1}$	True and estimated consecutive node distances between nodes i and $i+1$
$w_{i,j}$, $w_{i,i+1}$	Weights of $\hat{d}_{i,j}$ and $\hat{d}_{i,i+1}$
$n_{i,i+1}$	Number of detected steps between nodes i and $i+1$
k_m, b_m	Stride parameters for trace m
$E_{i,j}$	Potential energy of the spring attached to nodes i and j
$\overrightarrow{f_{i,j}}$	Elastic force exerted on node i from node j
$\mathcal{N}(i)$	Neighboring nodes of node i
d_{neigh}	Parameter for determining the neighboring nodes of a node
δ	Step size for updating the coordinates of nodes

of the 3-axis accelerometer signals for step detection only. To recover the true periodicity of the signal, we first apply a low-pass filter with a cut-off frequency set to 3 Hz to remove high-frequency noise and spikes. Then, we search for peaks to identify distinct steps. To reduce fake steps caused by accidental bouncing of the mobile device, we enforce two heuristic constraints:

- Minimum and maximum time duration of a step, e.g., 0.5 seconds and 1 second;
- Minimum and maximum changes in acceleration magnitudes during one step, e.g., 0.2 g and 2 g.

Existing works [25], [26] have shown that the stride model for converting steps to distances can vary greatly across individuals due to weights and heights, and even for the same individual due to physical exertion, ground and shoe types, hurriedness and health status, etc. We adopt the following walking distance model

$$d_{i,i+1} = k_m n_{i,i+1} + b_m, \qquad \forall m \tag{1}$$

where $n_{i,i+1}$ is the number of steps detected between consecutive nodes i and $i+1$ of trace m, and k_m and b_m are stride parameters to be estimated.

B. Junction Detection

The objective of junction detection is to identify nodes that we believe are taken from the same location and to set their distances to zero. As WiFi signals can vary greatly, it is generally hard to individually match two $WiFi\ fingerprints$, which are vectors of RSSI from nearby APs. Many approaches [11], [27], [28], [29] have turned to aligning sequences of rather than individual fingerprints. Among them, Smith-Waterman algorithm [27] that is guaranteed to find the optimal local alignment has made a success. However, we cannot apply the Smith-Waterman algorithm straightforwardly due to the following two reasons. First, users may go back and forth, making the alignment not monotone. Second, users may traverse the same path in opposite directions. Hence, we need to align each pair of sequences twice. To address these two issues, we follow [29] and employ a sliding-window technique for aligning $seqA$ and $seqB$.

Let $seqA = \mathbf{a}_1, \ldots, \mathbf{a}_m$ and $seqB = \mathbf{b}_1, \ldots, \mathbf{b}_n$ be two WiFi fingerprint sequences to be aligned, where $\mathbf{a}_i$ and $\mathbf{b}_j$ are the ith fingerprint of $seqA$ and jth fingerprint of $seqB$, m and n are their lengths. To reduce the comparison effort, we first extract WiFi fingerprint sequences with high AP overlap. We measure the AP overlap by the Jaccard score

$$J(\mathcal{A}, \mathcal{B}) = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}, \tag{2}$$

where $\mathcal{A}$ and $\mathcal{B}$ are the sets of hearable APs for $seqA$ and $seqB$ respectively.

Then, we split $seqB$ into k non-overlapping windows $seqB_1, \ldots, seqB_k$ with window size w and apply the Smith-Waterman algorithm for aligning $seqA$ and $seqB_i$. Now, we design the scoring scheme for the Smith-Waterman algorithm. Let $\mathbf{a}_i$ and $\mathbf{b}_j$ be WiFi fingerprints. We measure their similarity by the cosine similarity

$$\cos(\mathbf{a}_i, \mathbf{b}_j) = \frac{\mathbf{a}_i^T \mathbf{b}_j}{\|\mathbf{a}_i\| \|\mathbf{b}_j\|}. \tag{3}$$

For scoring, we fill a substitution matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$ by the following rule

$$\mathbf{S}_{i,j} = \begin{cases} 3, & \text{if } 1 - \cos(\mathbf{a}_i, \mathbf{b}_j) \leq \theta_1, \\ -1, & \text{if } 1 - \cos(\mathbf{a}_i, \mathbf{b}_j) > \theta_1 \end{cases} \tag{4}$$

where two fingerprints $\mathbf{a}_i$ and $\mathbf{b}_j$ are considered matched if the dissimilarity is smaller than the predefined threshold θ_1. In (4), a match has a reward of 3, and a mismatch has a penalty of -1. We further use a linear gap penalty that has the same score for opening and extending a gap, i.e., an insertion or a deletion. The linear gap penalty simplifies the scoring process and is defined as

$$\eta_k = k\eta_1, \tag{5}$$

where η_k is the penalty of a gap of length k and η_1 is the penalty of a single gap. In our setting, $\eta_1 = -1$.

With the scoring scheme defined, we now fill a scoring matrix $\mathbf{H} \in \mathbb{R}^{(m+1) \times (n+1)}$. We initialize the first row and column with zero and fill the matrix by the following rules

$$\mathbf{H}_{i,j} = \max \begin{cases} \mathbf{H}_{i-1,j-1} + \mathbf{S}_{i,j}, \\ \mathbf{H}_{i-1,j} + \eta_1, \\ \mathbf{H}_{i,j-1} + \eta_1, \\ 0 \end{cases} \tag{6}$$

where the diagonal move means a substitution, i.e., a match or a mismatch. The horizontal and vertical moves mean gaps.

After filling the matrix, we find the maximum score in the matrix and trace back the path of the previous maximum scores among horizontal move, vertical move, and diagonal move until a 0 is encountered. The path is the aligned indices $indA_i$ and $indB_i$. Note that it is possible that we can find several positions with the same maximum score in the scoring matrix. In that case, we randomly choose one as our alignment result. If the

Algorithm 1: mSW Algorithm.

Input : $seqA, seqB, \theta_1, \theta_2$;
Output : $\hat{indA}, \hat{indB}$;
$indA = \{\}$;
$indB = \{\}$;
for each window $seqB_i$ in $seqB$ **do**
  //Smith-Waterman algorithm
  Fill the substitution matrix $\mathbf{S}$ by (5);
  Initialize $\mathbf{H}'$s first row and first column with **0**;
  Fill the scoring matrix $\mathbf{H}$ **by** (6);
  Obtain alignment $indA_i, indB_i$ by tracing back$\mathbf{H}$;
  //Record medians of the alignment
  if $indB_i$ satisfies consistency condition (7) **then**
    Add med$(indA_i)$ to $indA$;
    Add med$(indB_i)$ to $indB$;
  end if
end for
//Interpolation
for i = 0: n-2 **do**
  $len = \max(indA_{i+1} - indA_i, indB_{i+1} - indB_i)$
  Add linspace$(indA_i, indA_{i+1}, len)$ to $\hat{indA}$
  Add linspace$(indB_i, indB_{i+1}, len)$ to $\hat{indB}$
endfor
$\hat{indA} =$ round$(\hat{indA})$
$\hat{indB} =$ round$(\hat{indB})$

aligned indices $indB_i$ in the ith window satisfy the consistency condition

$$\frac{\text{len}(indB_i)}{\text{len}(seqB_i)} \geq \theta_2, \tag{7}$$

it means most of $seqB_i$ is aligned with $seqA$. We then consider $indA_i$ and $indB_i$ as matched. We further add the medians of all valid aligned indices as $(\text{med}(indA_i), \text{med}(indB_i))$ to ind_i. As the last step, we evenly interpolate the indices $indA$ and $indB$ to obtain the final alignment. We summarize the steps in Algorithm 1.

1) Computation Complexity: To analyze the time complexity of the modified Smith-Waterman algorithm, we analyze the main steps of the algorithm. Let m and n be the length of $seqA$ and $seqB$, and $m > n$. Let w be the length of window $seqB_i$. The number of iterations is $\lceil \frac{n}{w} \rceil$. We aim to align $seqB_i$ with $seqA$ in each iteration. Computing the substitution matrix has a time complexity of $O(m)$. Filling the scoring matrix has a time complexity of $O(m)$. In the traceback step, finding the maximum score in the scoring matrix has a time complexity of $O(m)$. Hence, the time complexity of each iteration is $O(m)$. The total time complexity of the algorithm is $\lceil \frac{n}{w} \rceil \times O(m) = O(mn)$.

The algorithm fills a single matrix of size mw and stores at most $O(w)$ positions for the traceback in each iteration. To obtain the final alignment, it uses at most $O(m)$ in the interpolation step. Hence, the total space complexity of this algorithm is $O(m)$.

C. Uncertainty-Aware Distance Estimation

This section presents our uncertainty-aware neural network, UDENet, for estimating non-consecutive node distances based on WiFi signals. These distances are not exploited in existing works and they help by addressing the sparsity problem in the pairwise distance matrix. We will describe the selected RSSI features, followed by the network architecture.

1) RSSI Features: Given two WiFi fingerprints $\mathbf{fp}_i \in \mathbb{R}^{|\mathcal{A}|}$ and $\mathbf{fp}_j \in \mathbb{R}^{|\mathcal{A}|}$ observed at nodes i and j, where $\mathcal{A}$ is the set of all APs observed at the site, our goal is to estimate the physical distance between the two nodes using WiFi fingerprints. We note RSSI is affected by human activity and environment, i.e., the presence of walls or obstacles, in addition to distances [30]. Hence, we propose a set of features, i.e., RSSI difference, RSSI distance, and RSSI variation to estimate distances.

For nodes i and j, we select the k strongest APs heard by either i or j to create the set of APs $\mathcal{A}_k$. The APs with strong signals are useful for estimating physical distances. In this work, we follow [31] and set the RSSI of APs who are not heard by the node to -80 dBm. We set k to 10. We calculate the MAE, MSE, and euclidean distance (ED) over $\mathcal{A}_k$ as RSSI distance

$$\text{MAE}(\mathbf{fp}_i, \mathbf{fp}_j, \mathcal{A}_k) = \frac{1}{k} \sum_{a \in \mathcal{A}_k} |\mathbf{fp}_i(a) - \mathbf{fp}_j(a)|, \tag{8}$$

$$\text{MSE}(\mathbf{fp}_i, \mathbf{fp}_j, \mathcal{A}_k) = \frac{1}{k} \sum_{a \in \mathcal{A}_k} |\mathbf{fp}_i(a) - \mathbf{fp}_j(a)|^2, \tag{9}$$

$$\text{ED}(\mathbf{fp}_i, \mathbf{fp}_j, \mathcal{A}_k) = \sqrt{\sum_{a \in \mathcal{A}_k} |\mathbf{fp}_i(a) - \mathbf{fp}_j(a)|^2}, \tag{10}$$

where $|\mathbf{fp}_i(a) - \mathbf{fp}_j(a)| = 80$ if $\mathbf{fp}_i(a) = -80$ and $\mathbf{fp}_j(a) = -80$.

We further use absolute difference (AD) over $\mathcal{A}_k$ as RSSI difference

$$\text{AD}(\mathbf{fp}_i, \mathbf{fp}_j, \mathcal{A}_k) = |\mathbf{fp}_i(\mathcal{A}_k) - \mathbf{fp}_j(\mathcal{A}_k)|, \tag{11}$$

where $\mathbf{fp}_i(\mathcal{A}_k)$ is the fingerprint of selected k APs at node i. $|\cdot|$ is elementwise absolute value.

To account for the effect of human activities and environments, we propose RSSI variation features which include RSSI temporal variation and dual-band disparity. Human activity based on WiFi signals has been extensively studied [32], [33], [34], [35]. Signal strength changes can be leveraged to infer human activities roughly. The key observation is that different human activities can be recognized from the standard deviation of RSSI samples. Standard deviation is shown to be a relatively stable feature for recognizing human activities. We define temporal variation as the change in RSSI when a user moves from node i to $i+1$

$$\Delta_i(\mathcal{A}) = \text{MAE}(\mathbf{fp}_i, \mathbf{fp}_{i+1}, \mathcal{A}). \tag{12}$$

where we consider only all APs with RSSI greater than -80 dBm. When performing WiFi fingerprinting, considering only APs with RSSI greater than -80 dBm is a common practice because it helps to filter out weak and noisy signals that may lead to inaccurate location estimates. The sensitivity of our UDENet

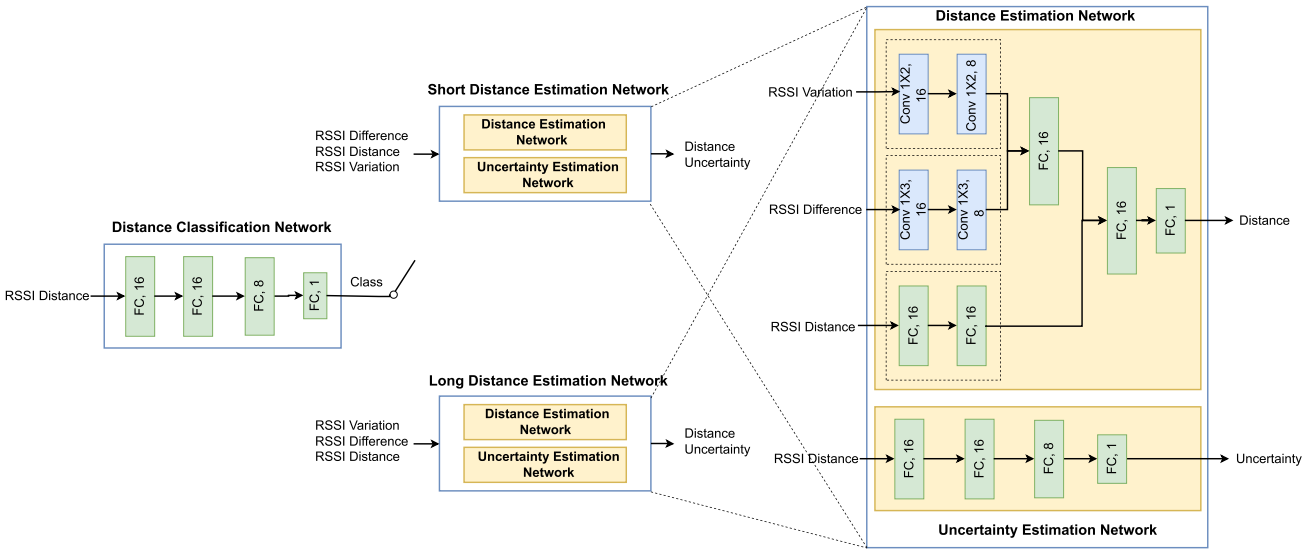


Fig. 3. Architecture of UDENet. FC denotes the fully connected layer. Conv denotes the convolutional layer.

to RSSI threshold depends on various factors, such as the environment and the type of tasks. Empirically, the performance of UDENet is not sensitive to the value of this parameter. We further define the difference in temporal variation at nodes i and j as

$$\Delta_{ij}(\mathcal{A}) = |\Delta_i(\mathcal{A}) - \Delta_j(\mathcal{A})|. \tag{13}$$

Signal propagation can be significantly affected by obstacles such as walls [36]. To account for the effect of walls, we harness the difference in the characteristics of the 2.4 and 5 GHz WiFi signals. The key observation is the difference in signal strengths between the 2.4 and 5 GHz signals when there is no wall is smaller than that when there is a wall. Hence, we define dual-band disparity which attempts to measure the difference between the temporal variation of the 2.4 GHz signal compared to the 5 GHz signal and calculate it as

$$\text{Env}_i = |\Delta_i(\mathcal{A}_{2.4\,G}) - \Delta_i(\mathcal{A}_{5\,G})|. \tag{14}$$

We further define the difference in the environment at nodes i and j as

$$\text{Env}_{ij} = |\text{Env}_i - \text{Env}_j|. \tag{15}$$

2) UDENet Architecture: Fig. 3 shows the architecture of UDENet. In UDENet, distance estimation is divided into two steps. First, we train a distance classification network to determine whether the distance between two nodes is a short or long distance. Then we train two separate networks for estimating short and long distances. The short distance network focuses on estimating distances when there are enough common APs in the two nodes. The long distance estimation network is trained to estimate distances when there are few common APs. Since it is well-known that errors in distance estimates cannot be captured by a simple additional white Gaussian noise (AWGN) model, along with each distance estimation network, we train a separate uncertainty estimation network to estimate the uncertainty in the distance estimate.

The distance classification model is a simple feed-forward network consisting of an input layer, three fully connected layers with a rectified linear unit (ReLU) activation function, and an output layer with a sigmoid activation function. The three fully connected layers consist of 16, 16, and 8 nodes respectively. We employ Adam as an optimizer and train the network to minimize the binary cross entropy.

The distance estimation network consists of three sub-networks. The first sub-network is a simple feed-forward network with two fully connected layers. It takes RSSI distance (i.e., MAE, MSE, and ED defined in (8-10)) as input. The feed-forward network is used here because RSSI distance is high-level information that is directly related to the pairwise distance. The second and third sub-networks are convolutional networks each consisting of two convolutional layers. They extract low-level information, i.e., RSSI difference and RSSI variation, to estimate the pairwise distance. We again employ Adam and train the network to minimize the mean squared error.

The uncertainty estimation network is a simple feed-forward network consisting of an input layer, three fully connected layers with ReLU activation function, and an output layer without ReLU. The fully connected layers have 16, 16, and 8 nodes respectively. We again employ Adam as an optimizer and train the network to minimize mean squared error.

We have trained UDENet using data from different environments separate from our experimental sites. We tested and confirmed that the distance estimates vary relatively little with the different training datasets.

V. Topological Map Construction

In this section, we describe how we use the detected junctions and estimated distances to construct a topological map and calibrate the stride parameters for all traces jointly. The estimated uncertainties of non-consecutive node distances by UDENet are transformed into weights and exploited in the map construction.

By using majorization minimization and block coordinate descend techniques, we develop an efficient algorithm to solve the topological map construction problem.

A. Formulation

We now formally state the topological map construction problem. Consider N nodes from M trajectories in a two-dimensional space. Our problem is to determine $\mathbf{X} \in \mathbb{R}^{N\times 2}$, the matrix of coordinates of all the nodes. The euclidean distance between nodes i and j is $d_{i,j}(\mathbf{X}) = \|\mathbf{X}_i - \mathbf{X}_j\|_2$. We have three types of information as inputs: consecutive node distances $\hat{d}_{i,i'}$, non-consecutive node distances $\hat{d}_{i,j}$ and node junctions. Consecutive node distances are estimated by the walking distance model in (1). Their confidences are $\hat{w}_{i,i'}$, which are treated as identical. Non-consecutive node distances and their uncertainties are estimated by UDENet. Their confidences are $\hat{w}_{i,j}$, which are reciprocals of uncertainties. The set of distinct consecutive node pairs (i, i') for the mth trace is denoted by $\mathcal{J}_m$. The set of distinct non-consecutive node pairs (i, j) is denoted by $\mathcal{I}$, The set of junction node pairs (i, j) is denoted by $\mathcal{K}$,

Motivated by the classical MDS algorithm, we estimate all node positions by solving the following stress function minimization problem

$$\mathcal{P}1: \min_{\mathbf{X},\mathbf{k},\mathbf{b}} \underbrace{\sum_{(i,j)\in\mathcal{I}} w_{i,j}(\hat{d}_{i,j} - d_{i,j}(\mathbf{X}))^2}_{\text{distance disparities for non-consecutive nodes}} + \underbrace{\sum_{m}\sum_{(i,i')\in\mathcal{J}_m} w_{i,i'}(\hat{d}_{i,i'}(k_m, b_m) - d_{i,i'}(\mathbf{X}))^2}_{\text{distance disparities for consecutive nodes}} \tag{16}$$

$$s.t. \quad \hat{d}_{i,i'} = k_m n_{i,i'} + b_m, \quad (i,i') \in \mathcal{J}_m, \quad \forall m \tag{17}$$

$$\mathbf{X}^T(\mathbf{e}_i - \mathbf{e}_j) = 0, \quad (i,j) \in \mathcal{K}. \tag{18}$$

Our formulation differs from the classical MDS algorithm in three ways: First, each intra-trace distance estimate is a function of k_m and b_m, which account for the step length and estimation noise for each trace. Our formulation enables us to learn node positions and the step length for each trace simultaneously. Previous formulations use a default step size and estimate only the node positions. Second, the weights $w_{i,j}$ and $w_{i,i'}$ are set to the assessed confidence of distance estimates. The confidence is inversely related to the uncertainty given by UDENet, preventing us from tedious parameter tuning. Third, the junction points detected by the mSW algorithm are imposed as a constraint, providing extra information and making our problem a constrained optimization problem. As a result, the classical MDS in previous formulations cannot be directly applied in our case. In our HMDS formulation, we need to optimize over two blocks of variables, and our optimization is a constrained optimization.

B. Algorithm

Motivated by iterative majorization (IM), we develop a surrogate function to approximate the original loss function. The central idea of the majorization method is to iteratively replace the original complicated function $f(x)$ with a simple surrogate function $g(x, z)$, where z in $g(x, z)$ is a fixed value and to generate a monotonically non-increasing sequence of function values. Following this idea, the two terms of our stress function have a similar form. For the first term, we expand and rewrite it as

$$\begin{aligned}\sigma_1(\mathbf{X}) &= \sum_{(i,j)\in\mathcal{I}} w_{i,j}(\hat{d}_{i,j} - d_{i,j})^2 \\ &= \sum_{(i,j)\in\mathcal{I}} w_{i,j}\hat{d}_{i,j}^2 + \sum_{(i,j)\in\mathcal{I}} w_{i,j}d_{i,j}^2 - 2\sum_{(i,j)\in\mathcal{I}} w_{i,j}\hat{d}_{i,j}d_{i,j} \\ &= \eta_1 + \zeta_1(\mathbf{X}) - 2\rho_1(\mathbf{X}),\end{aligned} \tag{19}$$

where η_1 is a constant term, $\zeta_1(\mathbf{X})$ is written in matrix form as

$$\begin{aligned}\zeta_1(\mathbf{X}) &= \text{tr}\left(\mathbf{X}^T\left(\sum_{(i,j)\in\mathcal{I}} w_{i,j}\mathbf{A}_{i,j}\right)\mathbf{X}\right) \\ &= \text{tr}(\mathbf{X}^T\mathbf{V}_1\mathbf{X}),\end{aligned} \tag{20}$$

where $\mathbf{A}_{i,j}$ is a matrix with $a_{i,i} = 1, a_{j,j} = 1, a_{i,j} = -1, a_{j,i} = -1$, and all other elements being zero. $\mathbf{V}_1$ is assumed to be irreducible, i.e., the directed graph associated with $\mathbf{V}_1$ is strongly connected. From the Cauchy-Schwarz inequality, the majorizor of $-\rho_1(\mathbf{X})$ is

$$\begin{aligned}-\rho_1(\mathbf{X}) &\le -\text{tr}\left(\mathbf{X}^T\left(\sum_{(i,j)} b_{i,j}\mathbf{A}_{i,j}\right)\mathbf{Z}\right) \\ &= -\text{tr}(\mathbf{X}^T\mathbf{B}_1(\mathbf{Z})\mathbf{Z}),\end{aligned} \tag{21}$$

where $\mathbf{Z}$ is a supporting point, and $\mathbf{B}_1(\mathbf{Z})$ has elements

$$b_{i,j} = \begin{cases} -\frac{w_{i,j}\hat{d}_{i,j}}{d_{i,j}(\mathbf{Z})}, & \text{if } i \ne j, d_{i,j}(\mathbf{Z}) \ne 0 \\ 0, & \text{if } i \ne j, d_{i,j}(\mathbf{Z}) = 0 \end{cases}$$

$$b_{i,i} = -\sum_{i\ne j} b_{i,j}. \tag{22}$$

In (21), the equality holds if $\mathbf{X} = \mathbf{Z}$. Combining (19), (20), and (21) gives us the majorizor of the first term in (16) as

$$\begin{aligned}\sigma_1(\mathbf{X}) &\le \eta_1 + \text{tr}(\mathbf{X}^T\mathbf{V}_1\mathbf{X}) - 2\text{tr}(\mathbf{X}^T\mathbf{B}_1(\mathbf{Z})\mathbf{Z}) \\ &= \tau_1(\mathbf{X};\mathbf{Z}).\end{aligned} \tag{23}$$

Similarly, we can obtain the majorizor of the second term in (16) as

$$\begin{aligned}\sigma_2(\mathbf{X},\mathbf{k},\mathbf{b}) &\le \eta_2(\mathbf{k},\mathbf{b}) + \text{tr}(\mathbf{X}^T\mathbf{V}_2\mathbf{X}) - 2\text{tr}(\mathbf{X}^T\mathbf{B}_2(\mathbf{k},\mathbf{b};\mathbf{Z})\mathbf{Z}) \\ &= \tau_2(\mathbf{X},\mathbf{k},\mathbf{b};\mathbf{Z}),\end{aligned} \tag{24}$$

where $\eta_2(\mathbf{k},\mathbf{b})$ is a function of $\mathbf{k} \in \mathbb{R}^M$ and $\mathbf{b} \in \mathbb{R}^M$. Now, we apply the method of Lagrange multipliers to find the local minima of a function subject to equality constraints. Let the majorizor of (16) be $\tau(\mathbf{X},\mathbf{k},\mathbf{b};\mathbf{Z}) = \tau_1(\mathbf{X};\mathbf{Z}) + \tau_2(\mathbf{X},\mathbf{k},\mathbf{b};\mathbf{Z})$, we write the Lagrangian as

$$\begin{aligned}\mathcal{L}(\mathbf{X},\mathbf{k},\mathbf{b},\mathbf{u};\mathbf{Z}) &= \tau(\mathbf{X},\mathbf{k},\mathbf{b};\mathbf{Z}) \\ &\quad + \sum_{(i,j)\in\mathcal{K}} \mathbf{u}_{(i,j)}^T\mathbf{X}^T(\mathbf{e}_i - \mathbf{e}_j),\end{aligned} \tag{25}$$

where $\mathbf{u}_{(i,j)} \in \mathbb{R}^2$ is the Lagrange multiplier associated with the equality constraint for junction nodes (i, j). We aim to find the stationary points of $\mathcal{L}$, which are the necessary conditions for the optimal solution corresponding to the original constrained problem [37]. However, the Lagrangian (25) is hard to optimize due to its nonconvexity. Fortunately, it is convex in each block of variables [38]. Specifically, $\mathcal{L}$ is quadratic in $\mathbf{X}, \mathbf{k}, \mathbf{b}$ and linear in $\mathbf{u}$. The saddle point of $\mathcal{L}$ can be computed efficiently by setting the partial derivatives of $\mathcal{L}(\mathbf{X}, \mathbf{k}, \mathbf{b}, \mathbf{u}; \mathbf{Z})$ to zero.

With $\mathbf{k}$ and $\mathbf{b}$ fixed

$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = 2\mathbf{V}\mathbf{X} - 2\mathbf{B}(\mathbf{k}, \mathbf{b}; \mathbf{Z})\mathbf{Z} + \sum_{(i,j)\in\mathcal{K}} (\mathbf{e}_i - \mathbf{e}_j)\mathbf{u}_{(i,j)}^T = 0, \tag{26}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}_{(i,j)}} = \mathbf{X}^T(\mathbf{e}_i - \mathbf{e}_j) = 0, \quad (i,j) \in \mathcal{K}. \tag{27}$$

where $\mathbf{V} = \mathbf{V}_1 + \mathbf{V}_2$, $\mathbf{B}(\mathbf{k}, \mathbf{b}; \mathbf{Z}) = \mathbf{B}_1(\mathbf{Z}) + \mathbf{B}_2(\mathbf{k}, \mathbf{b}; \mathbf{Z})$. Then, $\mathbf{X}$ and $\mathbf{u}$ can be obtained analytically by

$$\begin{bmatrix} \mathbf{X} \\ \mathbf{u}^T \end{bmatrix} = \begin{bmatrix} 2\mathbf{V} & \mathbf{E} \\ \mathbf{E}^T & 0 \end{bmatrix}^{\dagger} \begin{bmatrix} 2\mathbf{B}(\mathbf{k}, \mathbf{b}; \mathbf{Z})\mathbf{Z} \\ 0 \end{bmatrix}, \tag{28}$$

where $\mathbf{u} = [\ldots, \mathbf{u}_{(i,j)}, \ldots]$, $\mathbf{E} = [\ldots, \mathbf{e}_{(i,j)}, \ldots]$ and $\mathbf{e}_{(i,j)} = \mathbf{e}_i - \mathbf{e}_j$.

With $\mathbf{X}$ and $\mathbf{u}$ fixed, k_m and b_m can be obtained by setting the partial derivatives to zero as follows:

$$\frac{\partial \mathcal{L}}{\partial k_m} = \sum_{(i,i')\in\mathcal{J}_m} 2w_{i,i'}(k_m n_{i,i'} + b_m)n_{i,i'} - 2\sum_{(i,i')\in\mathcal{J}_m} w_{i,i'} n_{i,i'} d_{i,i'} = 0 \tag{29}$$

$$\frac{\partial \mathcal{L}}{\partial b_m} = \sum_{(i,i')\in\mathcal{J}_m} 2w_{i,i'}(k_m n_{i,i'} + b_m) - 2\sum_{(i,i')\in\mathcal{J}_m} w_{i,i'} d_{i,i'} = 0. \tag{30}$$

The above equations constitute a system of linear equations. By stacking k_m and b_m into vectors, we can solve $\mathbf{k}$ and $\mathbf{b}$ analytically by

$$\begin{bmatrix} \mathbf{k} \\ \mathbf{b} \end{bmatrix} = \Phi^{\dagger}\boldsymbol{\phi}, \tag{31}$$

where

$$\Phi_1 = \text{diag}\left(\left[\sum_{(i,i')\in\mathcal{J}_1} w_{i,i'} n_{i,i'}^2, \ldots, \sum_{(i,i')\in\mathcal{J}_M} w_{i,i'} n_{i,i'}^2\right]\right), \tag{32}$$

$$\Phi_2 = \text{diag}\left(\left[\sum_{(i,i')\in\mathcal{J}_1} w_{i,i'} n_{i,i'}, \ldots, \sum_{(i,i')\in\mathcal{J}_M} w_{i,i'} n_{i,i'}\right]\right), \tag{33}$$

Algorithm 2: HMDS Algorithm.

Input : $w_{i,j}, w_{i,i'}, \hat{d}_{i,j}, n_{i,i'}$;
Output : $\mathbf{X}, \mathbf{u}, \mathbf{k}, \mathbf{b}$;
1: Initialization: $\mathbf{X}_0$, $\mathbf{Z}_0$, $\mathbf{k}_0$, $\mathbf{b}_0$, $\mathbf{u}_0$;
2: $t \leftarrow 1$;
3: **while** convergence condition not reached **do**
4: $\quad \mathbf{X}_t, \mathbf{u}_t \leftarrow (28)$; // update coordinate
5: $\quad \mathbf{k}_t, \mathbf{b}_t \leftarrow (31)$; // update stride
6: $\quad \mathbf{Z}_t \leftarrow \mathbf{X}_t$;
7: $\quad t \leftarrow t + 1$;
8: **end while**

$$\Phi_3 = \text{diag}\left(\left[\sum_{(i,i')\in\mathcal{J}_1} w_{i,i'}, \ldots, \sum_{(i,i')\in\mathcal{J}_M} w_{i,i'}\right]\right), \tag{34}$$

$$\Phi = \begin{bmatrix} \Phi_1 & \Phi_2 \\ \Phi_2 & \Phi_3 \end{bmatrix} \in \mathbb{R}^{2M\times 2M}, \tag{35}$$

$$\boldsymbol{\phi} = \begin{bmatrix} \sum_{(i,i')\in\mathcal{J}_1} w_{i,i'} n_{i,i'} d_{i,i'} \\ \vdots \\ \sum_{(i,i')\in\mathcal{J}_M} w_{i,i'} n_{i,i'} d_{i,i'} \\ \sum_{(i,i')\in\mathcal{J}_1} w_{i,i'} d_{i,i'} \\ \vdots \\ \sum_{(i,i')\in\mathcal{J}_M} w_{i,i'} d_{i,i'} \end{bmatrix} \in \mathbb{R}^{2M}. \tag{36}$$

We summarize the HMDS algorithm in Algorithm 2

In the HMDS algorithm, we introduce the Lagrangian (25) to handle the constraints and the block coordinate descent (BCD) technique to optimize the two blocks of variables in turn - (28) for the first block and (31) for the second block. One advantage of BCD is that it can be easily parallelized - the updates to different blocks of variables can be computed independently and in parallel. This makes BCD particularly useful for large-scale optimization problems, where the ability to parallelize the computation can greatly reduce the time required to find a solution. Another advantage of BCD is that it can be used to solve problems with constraints since the variables can be updated in a way that respects the constraints.

C. Convergence Analysis

In this session, we equip the HMDS algorithm with the Armijo rule and give the convergence analysis of the HMDS algorithm. Let $\mathbf{x} = \text{vec}(\mathbf{X})$, $\mathbf{y} = [\mathbf{x}; \mathbf{k}; \mathbf{b}]$ and $\mathbf{d} = [\mathbf{d}^{\mathbf{x}}; \mathbf{d}^{\mathbf{k}}; \mathbf{d}^{\mathbf{b}}]$. Let $\bar{\mathbf{y}} = [\bar{\mathbf{x}}; \bar{\mathbf{k}}; \bar{\mathbf{b}}]$ be a limit point of the sequence $\{\mathbf{y}_t\}$. Let $\{\mathbf{y}_{t_i} | i = 0, 1, \ldots\}$ be a subsequence of $\{\mathbf{y}_t\}$ that converges to $\mathbf{y}^*$. For ease of discussion, we focus on the update of $\mathbf{x}$. In the tth iteration of HMDS algorithm, we minimize $\mathcal{L}$ in (25) with respect to $\mathbf{x}$ and obtain $\mathbf{x}$ by (28). We choose a feasible descent direction by $\mathbf{d}_t^{\mathbf{x}} = \mathbf{x} - \mathbf{x}_{t-1}$ and set $\mathbf{d}_t = [\mathbf{d}_t^{\mathbf{x}}; 0; 0]$. We further obtain a stepsize α_t by Armijo rule, which is defined as follows:

Given $\alpha_0 > 0$, $\beta \in (0,1)$ and $\sigma \in (0,1)$, we choose α_t to be the largest element in $\{\alpha_0\beta^n | n = 0, 1, \ldots\}$ such that

$$f(\mathbf{y}_{t-1}) - f(\mathbf{y}_{t-1} + \alpha_t \mathbf{d}_t) \geq -\sigma\alpha_t f'(\mathbf{y}_{t-1}; \mathbf{d}_t) \tag{37}$$

and

$$\mathbf{y}_{t-1} + \alpha_t \mathbf{d}_t \in \mathcal{X}, \tag{38}$$

where $f'(\mathbf{y}_{t-1}; \mathbf{d}_t)$ is the directional derivative of f at point $\mathbf{y}_{t-1}$ in the direction $\mathbf{d}_t$. The convergence of the HMDS algorithm is shown in Theorem 1.

Theorem 1. Every limit point of the iterates generated by the HMDS algorithm is a stationary point of $\mathcal{P}_1$.

Proof. Due to the space limit, we state the proof sketch here. Due to the use of the Armijo rule, we have

$$f(\mathbf{y}_t) - f(\mathbf{y}_{t+1}) \geq -\sigma\alpha_t f'(\mathbf{y}_t; \mathbf{d}_t) \geq 0. \tag{39}$$

Letting $t \to \infty$, we have

$$\lim_{t\to\infty} \alpha_t f'(\mathbf{y}_t; \mathbf{d}_t) = 0. \tag{40}$$

Since $\{f(\mathbf{y}_t)\}$ is monotonically non-increasing, we have

$$\lim_{t\to\infty} f(\mathbf{y}_t) = f(\bar{\mathbf{y}}), \tag{41}$$

where $\bar{\mathbf{y}}$ is a limit point.

Restricting to a subsequence $\{\mathbf{y}_{t_i}\}$ where $\mathbf{x}$ is updated, we follow the Theorem 4.1 (b) in [39] and prove that

$$\lim_{i\to\infty} \mathbf{d}_{t_i} = 0 \tag{42}$$

by contradiction.

Since $\mathbf{x}_{t_i} + \mathbf{d}_{t_i}$ is the minimizer of surrogate function $\tau(\mathbf{x}, \mathbf{y}_{t_i})$, we have

$$\tau(\mathbf{x}_{t_i} + \mathbf{d}_{t_i}; \mathbf{y}_{t_i}) \leq \tau(\mathbf{x}; \mathbf{y}_{t_i}), \quad \forall \mathbf{x} \in \mathcal{X}_1. \tag{43}$$

Combining (42) and (43) and letting $i \to \infty$, we have

$$\tau(\bar{\mathbf{x}}; \bar{\mathbf{y}}) \leq \tau(\mathbf{x}; \bar{\mathbf{y}}), \quad \forall \mathbf{x} \in \mathcal{X}_1. \tag{44}$$

Due to the first-order optimality condition and the fact that τ is a local approximation of f

$$\tau'(\mathbf{x}, \mathbf{y}; \mathbf{d}^{\mathbf{x}}) = f'(\mathbf{y}; \mathbf{d}), \forall \mathbf{d} = (\mathbf{d}^{\mathbf{x}}; 0; 0) \quad \text{with } \mathbf{x} + \mathbf{d}^{\mathbf{x}} \in \mathcal{X}_1, \tag{45}$$

we have

$$f'(\bar{\mathbf{y}}; \mathbf{d}) \geq 0, \forall \mathbf{d} = (\mathbf{d}^{\mathbf{x}}; 0; 0) \quad \text{with } \bar{\mathbf{x}} + \mathbf{d}^{\mathbf{x}} \in \mathcal{X}_1. \tag{46}$$

Moreover, since $\mathbf{d}_{t_i} \to 0$, we have

$$\lim_{i\to\infty} \mathbf{y}_{t_i+1} = \bar{\mathbf{y}}. \tag{47}$$

By restricting to the subsequence that $d_{t_i} \to 0$ and repeating the above argument for $\mathbf{k}$ and $\mathbf{b}$, we have

$$f'(\bar{\mathbf{y}}; \mathbf{d}) \geq 0, \quad \forall \mathbf{d} = (0; \mathbf{d}^{\mathbf{k}}; 0) \quad \text{with } \bar{\mathbf{k}} + \mathbf{d}^{\mathbf{k}} \in \mathcal{X}_2 \tag{48}$$

$$f'(\bar{\mathbf{y}}; \mathbf{d}) \geq 0, \quad \forall \mathbf{d} = (0; 0; \mathbf{d}^{\mathbf{b}}) \quad \text{with } \bar{\mathbf{b}} + \mathbf{d}^{\mathbf{b}} \in \mathcal{X}_3 \tag{49}$$

Using the regularity of f at point $\bar{\mathbf{y}}$ completes the proof. □

D. Computation Complexity

To analyze the time complexity of the HMDS algorithm, we essentially analyze each individual step of the algorithm. Let m and n be the number of traces and the number of positions. Usually, $n >> m$. The major computation in each iteration is calculating the pseudo-inverse using singular value decomposition (SVD). The complexity of SVD for solving $\mathbf{X}$ and $\mathbf{u}$ is $\mathcal{O}(n^3)$. The complexity of SVD for solving $\mathbf{k}$ and $\mathbf{b}$ is $\mathcal{O}(m^3)$. The stepsize search by the Armijo rule is of constant time complexity. Assuming the number of iterations until convergence is k, the total time complexity of the HMDS algorithm is $\mathcal{O}(kn^3)$.

VI. Map Rectification

In this session, we describe how we rectify the map obtained in the preceding section with reference to landmarks. We first find a transformation to roughly align the map with the landmarks. Then, we apply our AFD algorithm to rectify the map using landmarks.

A. Map Transformation

We aim to find the transformation matrices, including scaling, translation, rotation, and reflection, to align landmarks with the corresponding nodes in the topological map obtained in the preceding section. We obtain these transformation matrices by solving the following minimization problem

$$\mathcal{P}_2: \quad \min_{\mathbf{P}\in\text{Diag}(2),\mathbf{R}\in\text{SO}(2),\mathbf{t}\in\mathbb{R}^2} \|\mathbf{X}_{\text{dst}} - \mathbf{P}\mathbf{R}\mathbf{X}_{\text{src}} - \mathbf{t}\|_F^2 \tag{50}$$

where $\mathbf{X}_{\text{dst}}$ and $\mathbf{X}_{\text{src}}$ are the coordinates of landmarks and corresponding nodes, $\mathbf{P} \in \text{Diag}(2)$ is a two-dimensional scaling matrix which is also a diagonal matrix, $\mathbf{R} \in \text{SO}(2)$ is a two-dimensional rotation matrix and $\mathbf{t} \in \mathbb{R}^2$ is a translation vector. The difficulty in solving $\mathcal{P}_2$ is the constraint that $\mathbf{P}$ is a diagonal matrix. By relaxing the constraint to $\mathbf{P} \in \mathbb{R}^{2\times 2}$, we obtain the relaxed problem $\mathcal{P}_2'$ which is a manifold optimization problem. We employ the trust-region algorithm to solve $\mathcal{P}_2'$. The problem can be solved efficiently by off-the-shelf solvers, such as Manopt [40]. Then, we apply the obtained transformation matrices to obtain the transformed coordinates as

$$\mathbf{X}_{\text{trans}} = \mathbf{P}\mathbf{R}\mathbf{X} + \mathbf{t}, \tag{51}$$

where $\mathbf{X}$ is the coordinates of nodes obtained by our HMDS algorithm.

B. Algorithm

The AFD algorithm fixes the positions of the landmarks and moves other nodes in a 2D euclidean space such that the euclidean distances between nodes accurately fit the estimated distances. Let $\hat{d}_{i,j}$ be the estimated distance between nodes i and j, and $\mathbf{X}_i$ be the coordinates of node i. We can characterize the embedding error as

$$\begin{aligned} E &= \sum_{(i,j)} E_{i,j} \\ &= \sum_{(i,j)} (\|\mathbf{X}_i - \mathbf{X}_j\|_2 - \hat{d}_{i,j})^2. \end{aligned} \tag{52}$$

The embedding error is an analogue to the potential energy of a spring-mass system. Thus, minimizing the embedding error is equivalent to minimizing the energy of a spring-mass system.

The main idea of the AFD algorithm is to fix a few nodes to the landmarks and replace each edge in the graph with a spring to form a spring-mass system. Each node maintains its own current coordinates, starting with $\mathbf{X}$ from the HDMS algorithm. In each iteration, a node communicates with neighboring nodes, measures the distances to these nodes, and adjusts its current coordinates according to the elastic forces exerted on it. The elastic force exerted on node i from node j is defined as

$$\overrightarrow{\mathbf{f}_{i,j}} = c_1(\|\mathbf{X}_i - \mathbf{X}_j\|_2 - \hat{d}_{i,j})\overrightarrow{\mathbf{e}_{i,j}}, \tag{53}$$

where c_1 is the stiffness of the spring between nodes i and j, $\hat{d}_{i,j}$ is the rest length of the spring, and $\overrightarrow{\mathbf{e}_{i,j}} = \frac{(\mathbf{X}_j - \mathbf{X}_i)}{\|\mathbf{X}_j - \mathbf{X}_i\|}$ is the unit vector gives the direction of the force on node i.

While the minimum energy of the spring system corresponds to the minimum embedding error, it is not guaranteed that the global minimum can be obtained. To escape from trivial local minima, our AFD algorithm adopts an adaptive strategy to select the neighboring nodes to communicate in each iteration. The neighbor nodes of node i are

$$\mathcal{N}(i) = \{j | \hat{d}_{i,j} < d_{neigh}\}, \tag{54}$$

There is a tradeoff between increasing d_{neigh} for greater accuracy and keeping it small for escaping from local minima. AFD algorithm starts with a small value for d_{neigh} to prevent nodes from getting stuck in local minima. Then, we increase d_{neigh} in each iteration until d_{neigh} reaches its predefined maximum value.

The net force exerted on node i is the sum of elastic forces from neighboring nodes

$$\overrightarrow{\mathbf{f}_i} = \sum_{j \in \mathcal{N}(i)} \overrightarrow{\mathbf{f}_{i,j}}. \tag{55}$$

The coordinates of node i is updated by

$$\mathbf{X}_i = \mathbf{X}_i + \delta \overrightarrow{\mathbf{f}_i}. \tag{56}$$

where δ is the step size. The rate of convergence is governed by the step size δ. Large values of δ cause the AFD algorithm to typically oscillate over low-energy valleys and fail to converge. Small values of δ cause the AFD algorithm slow convergence. To improve the convergence of the algorithm, we employ an adaptive cooling scheme [41] to allow the algorithm to change step size depending on the progress. This adaptive scheme is motivated by the trust region algorithm for optimization. Let $\gamma \in (0, 1)$ be a cooling factor. If the energy of the system decreases, the step size is unchanged. If the energy of the system continuously decreases, we start to enlarge the step size by $\delta = \frac{\delta}{\gamma}$. If the energy of the system increases, we shrink the step size by $\delta = \gamma\delta$.

We summarize the AFD algorithm in Algorithm 3.

Algorithm 3: AFD Algorithm.

Input : $d_{neigh}, \epsilon_1, \hat{d}_{i,j}, \hat{d}_{i,i'}, \gamma$;
Output : $\mathbf{X}$;
$\mathbf{X}_0 \leftarrow \mathbf{X}_{\text{trans}}$;
$\Delta\mathbf{X} \leftarrow$ inf;
$E_0 \leftarrow$ inf;
$t \leftarrow 1$;
while $\|\Delta\mathbf{X}\|_F > \epsilon_1$ and $t < max\_iter$ **do**
  for $i \in \mathcal{U}$ **do**
    Compute $\mathcal{N}(i)$ using d_{neigh};
    $\overrightarrow{\mathbf{f}_i} \leftarrow (55)$;
  end for
  for $i \in \mathcal{U}$ **do**
    $\mathbf{X}_i \leftarrow (56)$;
  end for
  $t \leftarrow t + 1$;
  $d_{neigh} \leftarrow d_{neigh}(1 + 0.001t)$;
  $E_t \leftarrow (52)$;
  $\Delta E = E_t - E_{t-1}$;
  $\Delta\mathbf{X} = \mathbf{X}_t - \mathbf{X}_{t-1}$;
  if $\Delta E < 0$ **then**
    $progress \leftarrow progress + 1$;
    if $progress > 5$ **then**
      $progress \leftarrow 0$;
      $\delta \leftarrow \frac{\delta}{\gamma}$;
    end if
  else
    $progress \leftarrow 0$;
    $\delta \leftarrow \gamma\delta$;
  end if
end while
return $\mathbf{X}$;

C. Computation Complexity

The AFD algorithm has $O(n^2)$ running time, where n is the number of nodes in the input graph. In each iteration, each node communicates with neighboring nodes, and their elastic forces need to be computed, which is $O(dn)$, where $d = \max\{\deg(u) | u \in \mathcal{U}\}$ is the maximum of nodes' degree. Assuming the number of iterations to convergence is estimated to be k, the total time complexity of AFD algorithm is $O(kn)$. In addition, k is empirically proportional to n. Note that when combined with a multilevel approach, AFD can handle graphs with millions of nodes. Furthermore, it requires $O(dn)$ for storage of pairwise node distance.

VII. Experiment and Result

A. Datasets and Equipments

In this section, we compare the performance of our system against several existing systems. We collect traces of WiFi fingerprints at two sites: a campus and a shopping mall. Data is collected via a smartphone using an application we developed. The smartphone is Samsung Galaxy S8+ with Android version 8.0. To obtain the ground truth of node positions, the surveyor designs a walking path and puts checkpoints on the path. Checkpoints typically contain the starting point, ending point, and turning points whose locations are known by referring to the

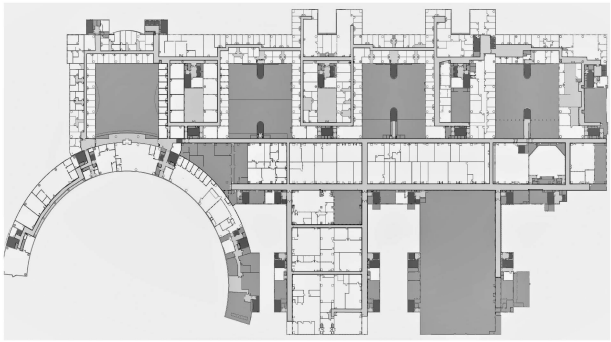


Fig. 4. Floor plan (campus).

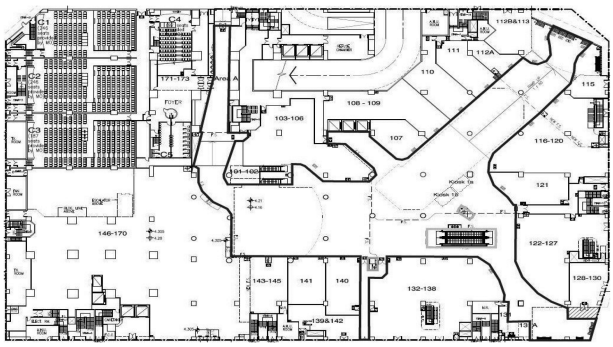


Fig. 5. Floor plan (mall).

TABLE II
STATISTICS OF THE DATASETS FOR RADIO MAP CONSTRUCTION

Site	Area	# Traces	# Nodes	# APs
Campus	350m×140m	105	2747	618
Mall	80m× 70m	61	1176	276

floor plan. Then, the surveyor follows the designed path and visits the checkpoints in order. When the user visits a checkpoint, she presses the button on the app to record the time of her visit. We can easily estimate the average walking velocity between consecutive checkpoints and generate a sequence of nodes with a time interval being 2 seconds. Since we know the positions of all nodes, we randomly select some of them as landmarks whose positions are known. In a real deployment, we can install BLE beacons that can broadcast their position at the site.

To imitate crowdsourced data, the surveyor collects data one trace at a time by walking in a normal fashion over a planned route. The app collects a variety of sensor readings, including WiFi RSSI values, accelerations, angular velocities, and magnetic fields, at their highest sampling frequencies. The acceleration readings are used to detect steps and determine the ground truth of step length. We synchronize different sensor readings to 0.5 Hz. The number of collected fingerprints is 2747 for the campus and 1176 for the mall. The average time interval between consecutive fingerprints is 2 seconds. Note that some areas on the floor plan may not be accessible. Figs. 4 and 5 show the floor plans of the two sites and Table II summarizes the two datasets.

Experiments are run on our laptop computer with Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHZ. The operating system is Windows 10. All the code is implemented in Python.

B. Baselines

We implemented the following baseline CIPSs for comparison:

- Zee which has two key components: Placement Independent Motion Estimator (PIME) and Augmented Particle Filter (APF). PIME uses mobile sensors such as accelerometer, compass, and gyroscope to estimate user motion. APF uses the motion estimates from PIME and the floor map to track the user's location on the floor.
- UnLoc which combines three components - dead-reckoning, urban sensing, and WiFi-based partitioning - into a framework for unsupervised localization. Both Seeded Landmarks (SLMs) from the floor plan and Organic Landmarks (OLMs) discovered from the collected data are used to improve the dead-reckoning for subsequent users, which in turn improves the location estimates of the SLMs/OLMs themselves. This circular process pushes the entire system to better accuracy over time.
- LiFs which transforms floor plan to stress-free floor plan by MDS and then creates a distance matrix between every pair of two consecutive fingerprints by step counter and completes the distance matrix by Floyd-Warshall algorithm. After using MDS to construct a fingerprint space, LiFS detects doors as landmarks to map fingerprints to locations. Since doors are not used in our scenario, to adapt to our experiments, we manually label all the landmarks.
- WiFi-RITA which generates selected user trajectories by PDR. WiFi-RITA merges the user trajectories by finding planar rotation and translation with reference to detected WiFi marks. After the trace merging, WiFi-RITA removes the outlier traces based on signal marks and landmarks.
- GraphIPS which dynamically generates accurate radio maps by utilizing crowdsourced smartphone WiFi and IMU data. GraphIPS fuses the crowdsourced data into a graph-based formulation and applies the MDS algorithm to compute the positions of users' steps. GraphIPS assumes AP locations are known and WiFi AoA data is available, both being not true in our scenario. To adapt to our experiments, we manually create all the virtual APs.

We summarize these CIPSs in Table III. The same data is used to evaluate the above CIPSs.

C. Map Accuracy

We study the map accuracy of Leto compared with the baseline CIPSs. Figs. 6 and 9 show the true radio maps of the campus and mall. Each color represents an individual trace. The same color may represent different traces for the simplicity of colors. Figs. 7 and 10 visually show the respective estimated radio maps by Leto. Figs. 8 and 11 show the CDFs of position errors of the estimated radio maps. From the results, we can observe that:

(1) Leto achieves better accuracy for both sites compared to most IPSs. The reason is that Leto can learn the topology of the

TABLE III
COMPARISON OF CIPSS

System	WiFi RSSI	Acc.	Gyro.	Mag.	Map	Landmark	AoA	Technique
Zee	✓	✓	✓	✓	✓	✗	✗	PDR, PF
Unloc	✓	✓	✓	✓	✓	Many landmarks	✗	PDR
LiFS	✓	✓	✗	✗	✓	All doors	✗	MDS, graph alignment
WiFi-RITA	✓	✓	✓	✗	✗	Many WiFi marks	✗	PDR, trace merging
GraphIPS	✓	✓	✓	✓	✗	All APs	✓	MDS
Leto	✓	✓	✗	✗	✗	Few landmarks	✗	HMDS, AFD

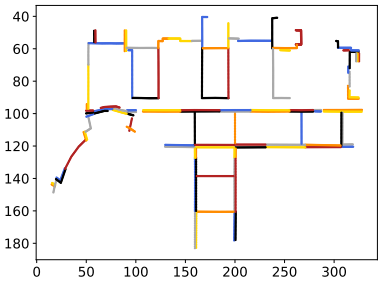


Fig. 6. True radio map (campus).

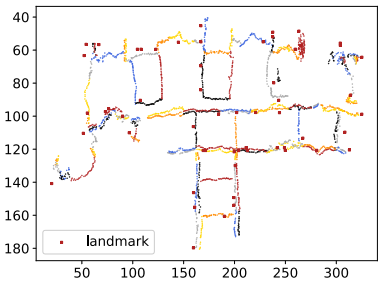


Fig. 7. Estimated radio map with 60 landmarks (campus).

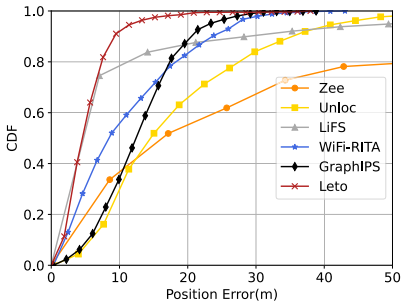


Fig. 8. CDF of position errors (campus).

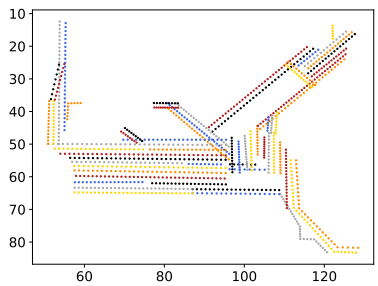


Fig. 9. True radio map (mall).

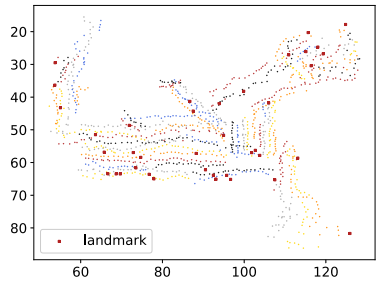


Fig. 10. Estimated radio map with 40 landmarks(mall).

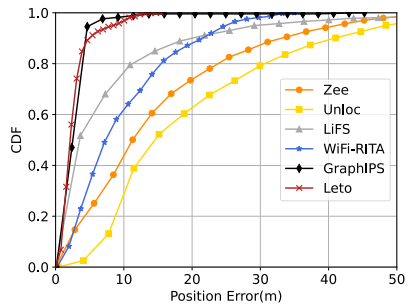


Fig. 11. CDF of position errors (mall).

radio map first and then make use of anchor points to calibrate the radio map, while Zee and Unloc construct the radio map trace by trace. It verifies that the topology is easy to learn and can help us better construct the radio map. LiFS and GraphIPS also learn the topology. However, LiFS associates the map with the floor plan corridor by corridor, which diminishes the benefits of learned topology. The rigid transformation used in LiFS is not stable for associating corridors with the floor plan. Note that GraphIPS achieves similar accuracy as ours on the mall dataset. The reason is that GraphIPS uses AOA data, which is not available in practice and GraphIPS generates them by simulated data. Also, GraphIPS uses AP locations as indirect anchor points, whose accuracies vary greatly in different sites.

(2) Leto is more deployable compared to the baseline CIPSs. From Table III, we use fewer sensors and prior knowledge. Most CIPSs use IMU signals for tracking user behavior, and Leto only uses the accelerometer signal for step detection. Zee, Unloc, and WiFi-RITA track user trajectories by PDR which may suffer from unstable user behavior. Most CIPSs assume a floor plan is available. Zee and LiFS heavily rely on the availability of a floor plan to map traces to. Unloc, LiFS, WiFi-RITA, and GraphIPS require enough landmarks, either automatically detected or

TABLE IV
STATISTICS OF THE DATASETS FOR LOCALIZATION EVALUATION

Site	Interval (days)	# Traces	# Nodes	# APs
Campus	91	32	549	362
Mall	91	20	235	185

determined by a site survey, to achieve satisfactory accuracies. For Unloc, each trace needs to observe at least one landmark. For LiFS, the junctions of corridors need to be correctly detected as landmarks. For WiFi-Rita, each trace needs to observe at least three WiFi marks. For GraphIPS, the AP locations are determined by a site survey.

(3) PDR-based CIPSs (i.e., Zee, Unloc, WiFi-RITA) generally have worse performance compared to graph-based CIPSs (i.e., Leto, GraphIPS, LiFS). The reason is that the IMU signal is noisy and the PDR result is unsatisfactory. Moreover, short traces contained in our datasets cause extra difficulty for PDR-based CIPSs. For Zee, short traces exhibit too few signatures for inferring their locations on the floor plan using a particle filter. For Unloc and WiFi-RITA, short traces usually reduce the probability of detecting enough landmarks to calibrate the PDR result.

(4) Accuracies of CIPSs on the mall dataset are generally better than that on the campus dataset. For graph-based CIPSs, like Leto and GraphIPS, the WiFi signal is utilized to infer pairwise distances, and there are higher probabilities for LOS (line of sight) at the mall, which is an open space. LOS can help improve the accuracy of pairwise distance estimation leading to improved radio map accuracy. For PDR-based CIPSs, like Zee, the campus map contains much more similar substructures, i.e., rectangles, than in a mall. It is harder for particle filters to converge to the correct locations, especially for short traces. Moreover, the area of the campus is much larger than that of the mall. Since we use the same number of particles for the two datasets, the mall is explored more fully than the campus.

D. Localization Accuracy

We study the localization accuracy of our radio map compared to the baseline CIPSs. We collected additional fingerprints at the mall and campus about three months later after the initial data collection. The true positions of nodes are generated as explained in Section 7.1. The information on datasets for localization evaluation is summarized in Table IV.

For a fair comparison, we adopt WKNN, which is a widely used localization technique. In this work, we set K to 3. From Figs. 18 and 19, we observe that:

(1) The performance of Leto is significantly better than the baseline models in terms of position errors. Leto's result is comparable to results obtained with site survey, especially on the mall dataset. Note that GraphIPS has a similar performance to ours on the mall dataset, which is consistent with the result of map construction errors. As shown in Figs. 18 and 19, 90 percent of localization errors are under 10 meters on the campus dataset, and 90 percent are under 5 meters on the mall dataset.

(2) The localization error is correlated with the map error. The localization error is slightly larger than the map error. The reason is that the fingerprints vary over time due to environmental changes - AP changes, unpredictable user behaviors, and so on.

TABLE V
PERFORMANCE OF THE DISTANCE CLASSIFICATION

Site	Accuracy	Precision	Recall	F1
Campus	0.925	0.934	0.975	0.954
Mall	0.922	0.993	0.937	0.964

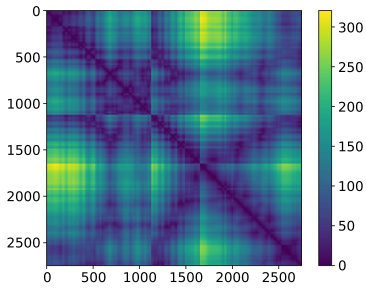


Fig. 12. True pairwise distances (campus).

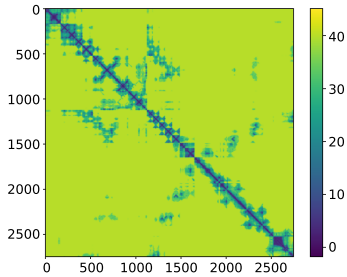


Fig. 13. Estimate pairwise distances (campus).

E. Pairwise Distance Estimation Accuracy

We study the accuracy of pairwise distance estimation. We divide the pairwise distance estimation into two steps. We first train a forward neural network to classify short and long distances. Then, we train two separate neural networks to estimate pairwise distances for each class. For the campus dataset, the short distance is from 0 to 10 meters, and the long distance is from 10 to 40 meters. For the mall dataset, the short distance is from 0 to 5 meters, and the long distance is from 5 to 40 meters. The results of distance classification are shown in Table V. From the results, we can observe that our model can classify different ranges with high accuracy.

The CDF of estimation errors is shown in Figs. 20 and 21. The ground truth and estimation of pairwise distance are shown in Figs. 12–17. We visualize the estimated pairwise distance matrix of campus and mall datasets in Figs. 13 and 16. The estimated distance is consistent with the ground truth shown in Figs. 12 and 15. Note that we only show distances under 40 meters. Our models only estimate pairwise distances under 40 meters due to the limited WiFi sensing range.

To demonstrate the robustness of our pairwise distance estimation model, we randomly add or remove some APs from the test data and evaluate the effect of AP changes on the accuracy

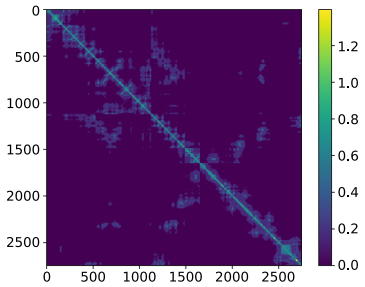


Fig. 14. Confidence of estimated distance (campus).

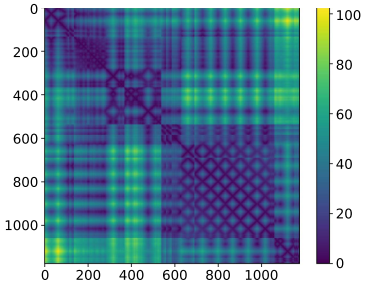


Fig. 15. True pairwise distances (mall).

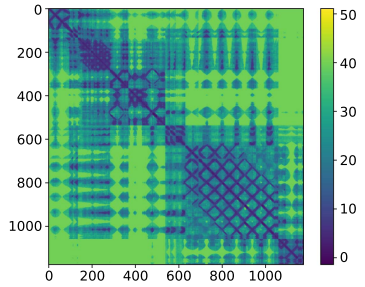


Fig. 16. Estimate pairwise distances (mall).

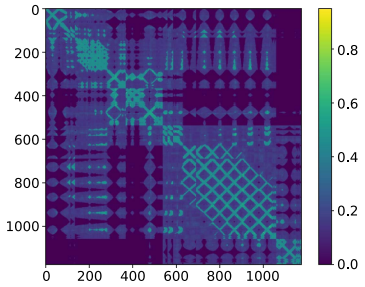


Fig. 17. Confidence of estimated distance (mall).

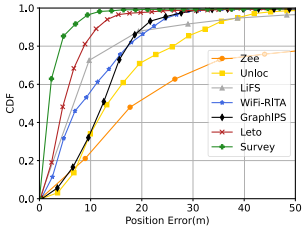


Fig. 18. CDF of localization errors (campus).

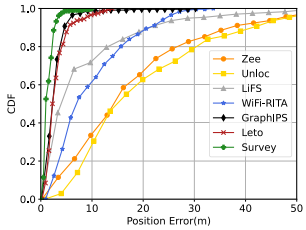


Fig. 19. CDF of localization errors (mall).

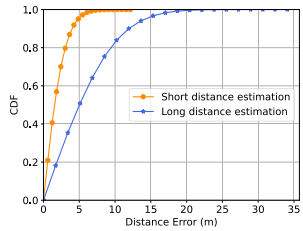


Fig. 20. CDF of distance errors (campus).

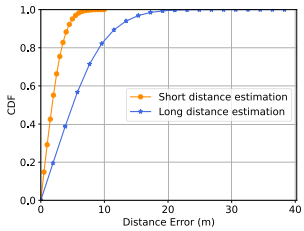


Fig. 21. CDF of distance errors (mall).

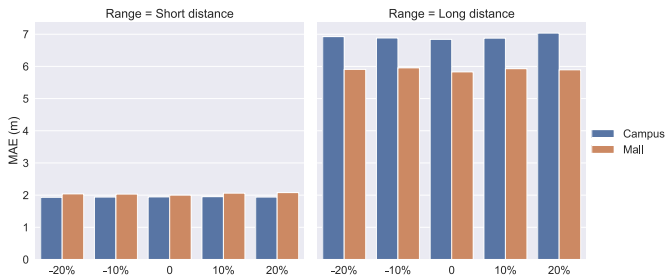


Fig. 22. Localization errors under AP changes.

of the estimates. The results are shown in Fig. 22. From the figure, we can observe that estimation errors change only slightly under different AP changes. The reason is that our AP selection criteria mitigate the effect of AP changes and crafted features are effective and robust under AP changes.

To demonstrate the generalization ability of our model, we increase the training dataset and evaluate the effect of training data size on the accuracy of pairwise distance estimation. Results are shown in Fig. 23. From the figure, we can observe that estimation errors change only slightly under different training sizes for short distances, while estimation errors are slightly reduced with increasing training size for long distances.

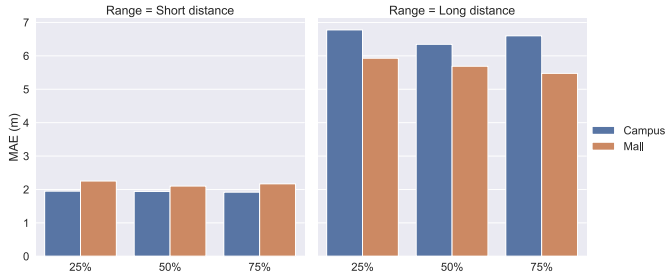


Fig. 23. Localization errors under different training sizes.

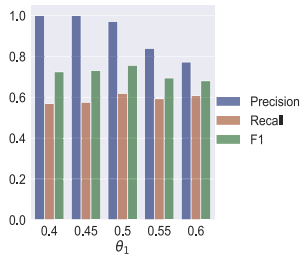


Fig. 24. Performance under different θ_1 (campus).

TABLE VI
PERFORMANCE OF THE JUNCTION DETECTION

Site	Precision	Recall	F1	Running time (s)
Campus	0.971	0.619	0.757	10.9
Mall	1	0.392	0.563	16.9

F. Junction Detection Accuracy

We study the performance of the junction detection algorithm. The true positions of all nodes are obtained, as explained in Section 7.1. When the distance between two nodes from different traces is smaller than 3 m, we consider them to be at the same location. As shown in Table VI, our junction detection algorithm achieves high precision on both datasets. It indicates our algorithm can detect junctions between traces reliably. Note that the recall of our algorithm is more than 50 % on the mall dataset. It indicates that our algorithm can discover many junctions that are not estimated accurately by our pairwise distance estimation model. This is because we utilize temporal information to align pairs of user traces. Moreover, our algorithm is highly efficient. It takes less than one minute to detect junctions among all traces on both datasets. The reason is that our sliding window technique and pruning strategies can greatly reduce the computation.

Our algorithm uses two heuristic thresholds θ_1 and θ_2 for determining fingerprint-level match and window-level match respectively. A smaller value of θ_1 means a stricter condition for a fingerprint-level match. Note that θ_1 for the campus data is empirically set larger than that for mall data. This is again because fingerprints exhibit higher similarity at the mall than at the campus. A Larger value of θ_2 means a stricter condition for a fingerprint-level match. To demonstrate the robustness of our algorithm, we investigate the effect of different values for these two thresholds. The results of θ_1 are shown in Figs. 24 and 25. We observe that the detection precision is near 1 over a certain range of θ_1 and deteriorates with larger values of θ_1. The detection recall differs at different sites. The recall slightly changes under different values of θ_1 on the campus dataset, while the recall significantly increases with increasing θ_1. It indicates that the algorithm is more sensitive to the θ_1 when it is applied to open space. The results of θ_2 are shown in Figs. 26 and 27. We observe that the detection performance is highly stable over a wide range of θ_2 on both datasets.

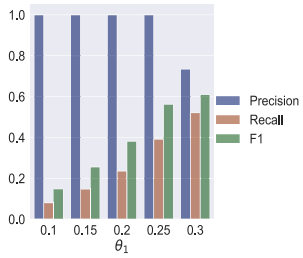


Fig. 25. Performance under different θ_1 (mall).

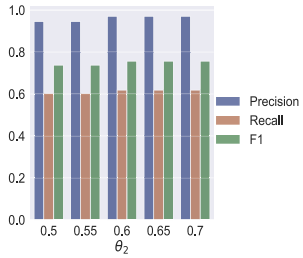


Fig. 26. Performance under different θ_2 (campus).

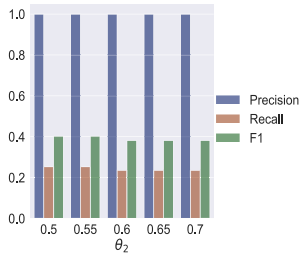


Fig. 27. Performance under different θ_2 (mall).

G. HMDS Performance

To demonstrate the effectiveness of our HMDS algorithm in constructing the radio map, we visualize the topology of the radio map in Figs. 28 and 29. From the figures, we can not directly evaluate the accuracy of the topological map, especially for Fig. 29. To show the accuracy of the constructed topological map, we need to refer to the result of map rectification as shown in Figs. 7 and 10. Since our adaptive force-directed (AFD) algorithm does not change the topology of the constructed map and it converges in a reasonable time, we can verify that the constructed topological map by HMDS has good accuracy and HMDS can effectively fuse the heterogeneous distance estimation to output a reasonable topology. We also show the MAE of the estimated

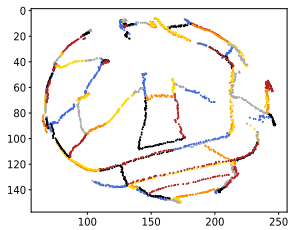


Fig. 28. Estimated topology of radio map (campus).

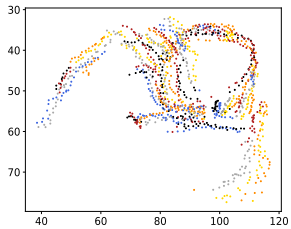


Fig. 29. Estimated topology of radio map (mall).

TABLE VII
PERFORMANCE OF THE STEP LENGTH ESTIMATION

Site	Method	MAE (m)
Campus	Constant	0.29
	HMDS	0.11
Mall	Constant	0.24
	HMDS	0.13

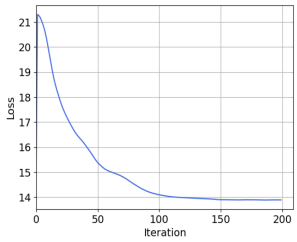


Fig. 30. Convergence of HMDS (campus).

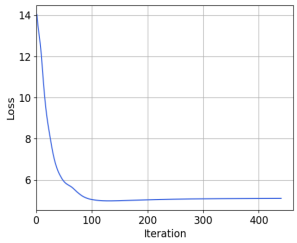


Fig. 31. Convergence of HMDS (mall).

step length compared to the constant method in Table VII. The constant method assumes each trace has a constant (0.76 m) step length.

To demonstrate the efficiency of HMDS, we evaluate the convergence and running time. Typical convergences of HMDS on different datasets are shown in Figs. 30 and 31. We observe monotone decreasing convergences on both datasets. It indicates the good convergence of HMDS. We also show the running time of HMDS in Table VIII. It indicates the low complexity of HMDS.

TABLE VIII
RUNING TIME OF THE MAP CONSTRUCTION

Site	# Nodes	Running time (min)
Campus	2747	5.93
Mall	1176	1.29

TABLE IX
PERFORMANCE OF THE MAP RECTIFICATION UNDER DIFFERENT NUMBERS OF LANDMARKS

Site	# Nodes	# Landmarks	MPE (m)
Campus	2747	20	11.48
		40	6.40
		60	5.40
		80	4.86
		100	4.34
Mall	1176	20	5.52
		40	2.47
		60	2.34
		80	2.18
		100	2.07

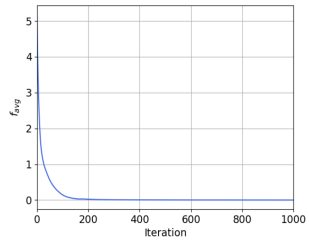


Fig. 32. Convergence of AFD (campus).

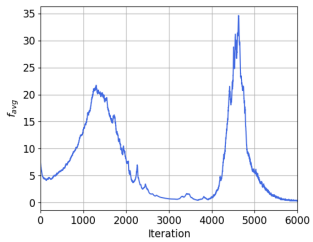


Fig. 33. Convergence of AFD (mall).

H. AFD Performance

In this work, we use 60 and 40 landmarks to rectify the radio map for campus and mall datasets. To demonstrate the effect of landmarks in rectifying the radio map, we further evaluate the MPE using different numbers of landmarks. The results are shown in Table IX. We observe that the accuracy of map rectification improves with the increasing number of landmarks.

To demonstrate the effectiveness of AFD in rectifying the radio map, we show the convergence of AFD in Figs. 32 and 33. We observe that the average force of nodes oscillates several times. It indicates our algorithm can escape from the local minima and converge to the global optimum by adaptively adjusting the communication range d_{neigh}. We show the running time of AFD in Table X. It indicates that AFD can find good solutions within a reasonable time.

TABLE X
RUNNING TIME OF THE MAP RECTIFICATION

Site	# Nodes	Running time (min)
Campus	2747	6.1
Mall	1176	11.3

VIII. CONCLUSION

In this paper, we propose Leto, a crowdsourced radio map construction scheme based on WiFi, accelerometer signals, and a few landmarks. Leto introduces several innovations for crowdsourced indoor radio map constructions. These innovations include enhancement to the graph-based CIPS formulation by introduction of individual stride parameters and combination of accelerometer-based and neural network-based distance estimations, a new low-complexity modified Smith-Waterman algorithm for sequence matching, a new low-complexity HMDS algorithm to replace MDS for stress minimization, and a new rectification step using an enhanced Adaptive Force Directed algorithm to better align the map with available landmarks. Our experiments show that the stride parameters can be estimated reasonably well which should in turn lead to better final results. Our modified Smith-Waterman algorithm is window-based and is capable of matching two sequences in the two directions at the same time. HMDS uses block coordinate descent to handle the two blocks of variables – node positions and stride parameters – in turn. We have also enhanced existing neural network-based distance estimation by introducing the concept of long and short-distance classification and uncertainty estimation. Our approach avoids the need for full IMU measurements, which can be sensitive to how mobile devices are carried.

We implemented Leto and conducted extensive experiments on our campus and a shopping mall. The results demonstrate that Leto significantly outperforms state-of-the-art CIPSs on the campus dataset, in terms of both the radio map accuracy and WKNN localization accuracy. For the mall data set, Leto is about at par with GraphIPS while both significantly outperform the others. We investigated our pairwise distance estimation accuracy, junction detection accuracy, HMDS performance with step length estimation accuracy, and AFD performance. We also investigated the running time and convergence of our algorithms. Leto requires little prior knowledge about the target site and consistently provides the best performance over two types of sites.

REFERENCES

[1] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. IEEE Conf. Comput. Commun. 19th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 2000, pp. 775–784.

[2] M. Youssef and A. Agrawala, "The Horus wlan location determination system," in *Proc. 3rd Int. Conf. Mobile Syst., Appl. Serv.*, 2005, pp. 205–218.

[3] A. Goswami, L. E. Ortiz, and S. R. Das, "Wigem: A learning-based approach for indoor localization," in *Proc. 7th Conf. Emerg. Netw. Experiments Technol.*, 2011, pp. 1–12.

[4] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 293–304.

[5] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Serv.*, 2012, pp. 197–210.

[6] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-Markie: Indoor pathway mapping made easy," in *Proc. 10th USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 85–98.

[7] S. Sorour, Y. Lostanlen, S. Valaee, and K. Majeed, "Joint indoor localization and radio map construction with limited deployment load," *IEEE Trans. Mobile Comput.*, vol. 14, no. 5, pp. 1031–1043, May 2015.

[8] C. Wu, Z. Yang, and Y. Liu, "Smartphones based crowdsourcing for indoor localization," *IEEE Trans. Mobile Comput.*, vol. 14, no. 2, pp. 444–457, Feb. 2014.

[9] S.-H. Jung, B.-C. Moon, and D. Han, "Unsupervised learning for crowdsourced indoor localization in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 11, pp. 2892–2906, Nov. 2016.

[10] X. Zhang, A. K.-S. Wong, C.-T. Lea, and R. S.-K. Cheng, "Unambiguous association of crowd-sourced radio maps to floor plans for indoor localization," *IEEE Trans. Mobile Comput.*, vol. 17, no. 2, pp. 488–502, Feb. 2017.

[11] J. Tan, H. Wu, K.-H. Chow, and S.-H. G. Chan, "Implicit multimodal crowdsourcing for joint RF and geomagnetic fingerprinting," *IEEE Trans. Mobile Comput.*, vol. 22, no. 2, pp. 935–950, Feb. 2023.

[12] X. Du, X. Liao, M. Liu, and Z. Gao, "CRCLoc: A crowdsourcing-based radio map construction method for WiFi fingerprinting localization," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12364–12377, Jul. 2022.

[13] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[14] A. Howard, M. J. Mataric, and G. Sukhatme, "Relaxation on a mesh: A formalism for generalized localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.. Expanding Societal Role Robot. Next Millennium*, 2001, pp. 1055–1060.

[15] M. Montemerlo et al., "FastSLAM: A factored solution to the simultaneous localization and mapping problem," *Proc. AAAI Nat. Conf. Artif. Intell.*, 2002, pp. 593–598.

[16] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *Int. J. Robot. Res.*, vol. 25, no. 5/6, pp. 403–429, 2006.

[17] B. Ferris, D. Fox, and N. D. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," in *Proc. Int. Joint Conf. Artif. Intell.*, 2007, pp. 2480–2485.

[18] R. Faragher and R. Harle, "SmartSLAM-An efficient smartphone indoor positioning system exploiting machine learning and opportunistic sensing," in *Proc. 26th Int. Tech. Meeting Satell. Division Inst. Navigation*, 2013, pp. 1006–1019.

[19] P. Mirowski, T. K. Ho, S. Yi, and M. MacDonald, "SignalSLAM: Simultaneous localization and mapping with mixed WiFi, Bluetooth, LTE and magnetic signals," in *Proc. Int. Conf. Indoor Positioning Indoor Navigation*, 2013, pp. 1–10.

[20] Y. Zhao, Z. Zhang, T. Feng, W.-C. Wong, and H. K. Garg, "GraphIPS: Calibration-free and map-free indoor positioning using smartphone crowdsourced data," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 393–406, Jan. 2021.

[21] Z. Li, X. Zhao, Z. Zhao, and T. Braun, "WiFi-RITA positioning: Enhanced crowdsourcing positioning based on massive noisy user traces," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3785–3799, Jun. 2021.

[22] C. Wu, Z. Yang, Y. Liu, and W. Xi, "WILL: Wireless indoor localization without site survey," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 4, pp. 839–848, Apr. 2013.

[23] W. Zhuo et al., "GRAFICS: Graph embedding-based floor identification using crowdsourced RF signals," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst.*, 2022, pp. 1051–1061.

[24] S. Mostafa, K. Harras, and M. Youssef, "A survey of indoor localization systems in multi-floor environments," 2022 *arXiv:20439648.v1*.

[25] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 421–430.

[26] W. Zijlstra, "Assessment of spatio-temporal parameters during unconstrained walking," *Eur. J. Appl. Physiol.*, vol. 92, no. 1, pp. 39–44, 2004.

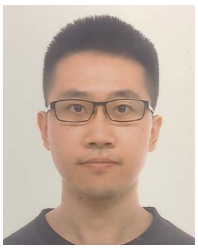
[27] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, 1981.

[28] K.-S. Lin, A. K.-S. Wong, T.-L. Wong, and C.-T. Lea, "Adaptive WiFi positioning system with unsupervised map construction," in *Proc. Int. Conf. Artif. Intell.*, 2015, pp. 636–642.

[29] H. Wu, S. He, and S.-H. G. Chan, "Efficient sequence matching and path construction for geomagnetic indoor localization," in *Proc. Int. Conf. Embedded Wireless Syst. Netw.*, 2017, pp. 156–167.

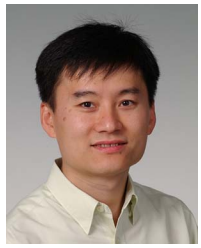
[30] T. He, J. Tan, W. Zhuo, M. Printz, and S.-H. G. Chan, "Tackling multipath and biased training data for IMU-assisted BLE proximity detection," 2022, *arXiv:2201.03817*.

[31] P. Sapiezynski, A. Stopczynski, D. K. Wind, J. Leskovec, and S. Lehmann, "Inferring person-to-person proximity using WiFi signals," *Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.*, vol. 1, no. 2, pp. 1–20, 2017.
[32] S. Sigg, M. Scholz, S. Shi, Y. Ji, and M. Beigl, "RF-sensing of activities from non-cooperative subjects in device-free recognition systems using ambient and local signals," *IEEE Trans. Mobile Comput.*, vol. 13, no. 4, pp. 907–920, Apr. 2014.
[33] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of WiFi signal based human activity recognition," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 65–76.
[34] H. Abdelnasser, M. Youssef, and K. A. Harras, "WiGest: A ubiquitous WiFi-based gesture recognition system," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 1472–1480.
[35] Y. Gu, F. Ren, and J. Li, "PAWS: Passive human activity recognition based on WiFi ambient signals," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 796–805, Oct. 2016.
[36] T. Nakatani, T. Maekawa, M. Shirakawa, and T. Hara, "Estimating the physical distance between two locations with Wi-Fi received signal strength information using obstacle-aware approach," *Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.*, vol. 2, no. 3, pp. 1–26, 2018.
[37] D. Kalman, "Leveling with lagrange: An alternate view of constrained optimization," *Math. Mag.*, vol. 82, no. 3, pp. 186–196, 2009.
[38] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imag. Sci.*, vol. 6, no. 3, pp. 1758–1789, 2013.
[39] P. Tseng and S. Yun, "A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training," *Comput. Optim. Appl.*, vol. 47, no. 2, pp. 179–206, 2010.
[40] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a matlab toolbox for optimization on manifolds," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1455–1459, 2014.
[41] Y. Hu, "Efficient, high-quality force-directed graph drawing," *Math. J.*, vol. 10, no. 1, pp. 37–71, 2005.



Yiwen Wang (Graduate Student Member, IEEE) received the bachelor of engineering degree from Southeast University, Nanjing, Jiangsu, China, in 2014, and the master of Philosophy degree in electronic and computer engineering from the Hong Kong University of Science and Technology (HKUST), in 2017. He is currently working toward the PhD degree with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (HKUST), Hong Kong, China. His research interest includes crowdsourced sensing, Internet of Things (IoT), and mobile computing.

Albert Kai-Sun Wong (Member, IEEE) received the SB, SM, EE, and PhD degrees in electrical engineering, all from the Massachusetts Institute of Technology, in 1982, 1984, 1984, and 1988, respectively. He joined the Hong Kong University of Science and Technology, in 2005 to support the University's Development in Nansha, China. Since 2008, he has been in the Department of Electronic and Computer Engineering and is currently a senior lecturer and associate head of the department. Prior, he was vice president, Wireless Communication Systems, of ASTRI, the Applied Science and Technology Research Institute of Hong Kong, and chief operating officer, Transtech Services Group, a company engaged in the building of a fiber preform plant. From 1988 to 2000, he was with AT&T and Lucent Technologies Bell Laboratories, where he was a member of technical staff, distinguished member of technical staff, technical Manager, Director of technical marketing, and Director, sales and technical marketing. He has also held visiting and adjunct faculty positions with the Chinese University of Hong Kong, Polytechnic University of New York, and Rutgers University. His current research interests include biomedical systems automation, wireless localization and tracking, and optical and data communication systems.



S.-H. Gary Chan (Senior Member, IEEE) received the BSE degree (highest honor) in electrical engineering from Princeton University (Princeton, NJ), with certificates in applied and computational mathematics, engineering physics, and engineering and management systems, and the MSE and PhD degrees in electrical engineering with a minor in business administration from Stanford University Stanford, CA, USA. He is currently a professor with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology (HKUST), Hong Kong. He is also affiliate professor in Innovation, Policy and Entrepreneurship Thrust Area of HKUST(GZ), chair of the Committee on Entrepreneurship Education Program, HKUST, and board director of Hong Kong Logistics and Supply Chain MultiTech R&D Center (LSCM). His research interest includes smart sensing and IoT, cloud and fog/edge computing, indoor localization and mobile computing, video/location/user/data analytics, and IT entrepreneurship. He has been an associate editor of *IEEE Transactions on Multimedia*, and a vice-chair of Peer-to-Peer Networking and Communications Technical Sub-Committee of IEEE Comsoc Emerging Technologies Committee. He has been guest editor of *ACM Transactions on Multimedia Computing, Communications and Applications*, *IEEE Transactions on Multimedia*, *IEEE Signal Processing Magazine*, *IEEE Communication Magazine*, etc. He is a steering committee member and was the TPC chair of IEEE Consumer Communications and Networking Conference (IEEE CCNC), and area chair of the multimedia symposium of IEEE Globecom and IEEE ICC.

Wai Ho Mow (Senior Member, IEEE) received the PhD degree in information engineering from the Chinese University of Hong Kong, in 1993. From 1997 to 1999, he was with the Nanyang Technological University, Singapore. He has been with the Hong Kong University of Science and Technology (HKUST), since 2000 and is currently a professor. He published two books and more than 220 journal/conference publications and is the inventor of 38 patents. His research areas include coding and information theory, wireless communications, optical camera communications, and thermographic signal processing. He pioneered the lattice approach to signal detection problems, including sphere decoding and complex lattice reduction-aided detection. He unified all known constructions of perfect roots-of-unity (aka CAZAC) sequences, which have been widely used as communication preambles and radar signals. His joint work won the top prizes of more than 10 project/paper competitions, including the 2014 HK U-21 IoT Gold Award for Revolutional Concept, the Best Paper Award of 2013 and the 2016 Asia-Pacific Communications Conference, and the Best Mobile App Award at ACM MobiCom'2013. His coinvented picture barcode PiCode was highlighted as one of the four local innovations in the 2015 International IT Fest organized by the Office of the Government Chief Information Officer, Hong Kong. He is a Past Chair of the Hong Kong Chapter of the IEEE Information Theory Society, and was the General/Program chair of six conferences, including SETA'2018 held in HK. He served on the Editorial Boards of six journals, including *IEEE Transactions on Wireless Communications*. He is a past member of the Radio Spectrum Advisory Committee, Office of the Telecommunications Authority, Hong Kong S.A.R. Government.