WiMU: Real-time Indoor Localization via Wi-Fi/IMU Fusion with Minimal Site Survey

QIRUI YANG, The Hong Kong University of Science and Technology, Hong Kong SAR
HUATAO XU, The Hong Kong University of Science and Technology, Hong Kong SAR
MENGXUAN SONG, The Hong Kong University of Science and Technology, Hong Kong SAR
MO LI*, The Hong Kong University of Science and Technology, Hong Kong SAR

With the widespread deployment of WiFi infrastructure, WiFi RSSI fingerprinting has been extensively explored for indoor localization. However, conventional fingerprinting approaches require labor-intensive data collection and are often constrained by limited sampling rates in practical deployments. To address these limitations, we propose WiMU, a real-time indoor localization system that integrates WiFi and inertial measurement unit (IMU) data to enhance the real-time performance and accuracy of localization. WiMU operates on commodity WiFi infrastructure without the need for additional hardware, leveraging crowdsourced user trajectories to learn spatial representations of access points (APs). These representations can be fine-tuned with minimal labeled data to support effective localization. Extensive evaluations show that WiMU outperforms three state-of-the-art methods, significantly reducing the overhead while maintaining high positioning accuracy. Notably, it achieves an average localization error of 5.247 meters using no more than 100 fingerprints with known locations. WiMU has also been successfully deployed in real-world environments, including a university campus and a warehouse, demonstrating its practical effectiveness for real-time indoor localization.

Additional Key Words and Phrases: Indoor Localization, Real-time Localization, WiFi RSSI Fingerprinting, Sensor Fusion, Graph Neural Network

1 Introduction

WiFi-based indoor localization has attracted considerable attention in recent years, driven by the widespread availability of WiFi infrastructure. A variety of WiFi signal features have been explored for localization purposes, including angle of arrival (AoA), time of flight (ToF), channel state information (CSI) fingerprints, and received signal strength indicator (RSSI) fingerprints. Among these, RSSI remains the only feature that can be extracted directly from commodity WiFi access points (APs) without requiring any hardware modifications [1]. While other signal features can theoretically provide higher accuracy, their practical adoption is limited due to the dependence on non-standard chipsets or protocols that are rarely supported in real-world deployments [1]. Consequently, RSSI fingerprinting remains the most feasible and cost-effective approach for large-scale WiFi-based indoor localization.

There are two primary challenges associated with the deployment of WiFi RSSI fingerprinting systems. First, the collection of RSSI fingerprints is labor-intensive [2]. Traditional localization approaches rely on collecting a large number of labeled RSSI fingerprints and estimating user positions by identifying the most similar fingerprints and averaging their associated locations [3–5]. With advances in machine learning (ML), numerous studies have employed neural networks to directly learn mappings from RSSI fingerprints to physical locations [6–9]. Nevertheless, both methods depend heavily on extensive site surveys to annotate RSSI data with precise location labels. For example, constructing a fingerprint database with a $5m \times 5m$ resolution for a six-story shopping mall covering $36{,}582m^2$ required three individuals working for ten days [1].

*Mo Li is the Corresponding author.

Authors' Contact Information: Qirui Yang, qyangau@connect.ust.hk, The Hong Kong University of Science and Technology, Hong Kong SAR; Huatao Xu, huatao@ust.hk, The Hong Kong University of Science and Technology, Hong Kong SAR; Mengxuan Song, msongae@connect.ust.hk, The Hong Kong University of Science and Technology, Hong Kong SAR; Mo Li, lim@ust.hk, The Hong Kong University of Science and Technology, Hong Kong SAR.

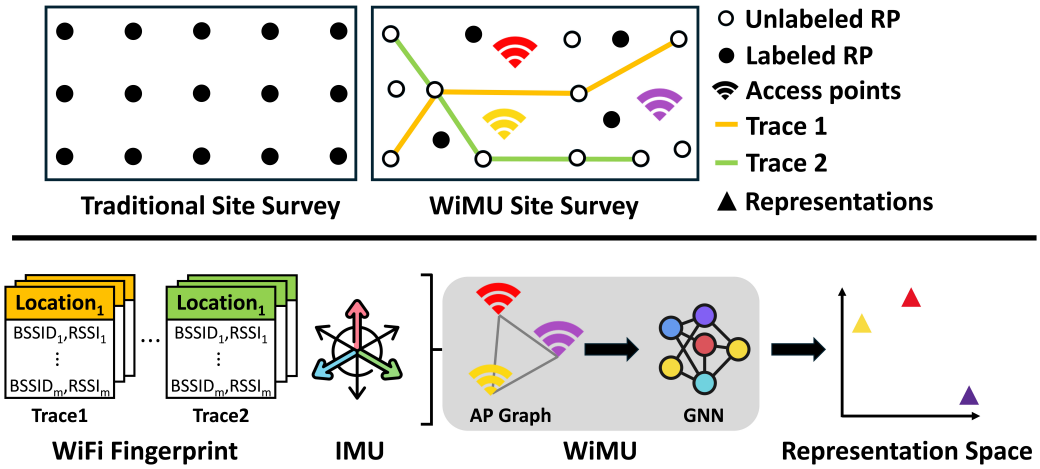


Fig. 1. Comparison between traditional and WiMU site survey settings. The lower half shows the key rationale of WiMU.

Secondly, **WiFi sampling frequency is inherently constrained**. On most smartphones, power-saving mechanisms limit RSSI sampling to approximately one measurement every 15 seconds; even under optimal conditions, the rate seldom exceeds one every 5 seconds [10]. This sparse sampling reduces temporal resolution, impeding real-time localization and making it challenging to reconstruct user trajectories between measurements using only WiFi data.

To address the above challenges, we propose WiMU, a lightweight real-time indoor localization system that fuses WiFi and IMU data from commodity mobile devices. WiMU tackles these issues through two key designs. Firstly, it employs a crowdsourced data collection mechanism that requires minimal manual effort (see Figure 1). As users move naturally through indoor spaces, their smartphones passively and periodically record WiFi and IMU signals. This enables large-scale data collection without active user involvement. The locations where WiFi RSSI fingerprints are captured along user trajectories are termed reference points (RPs). Since the data is collected passively, the precise locations of both RPs and APs are unknown, referred to as unlabeled RPs and unlabeled APs. Although often overlooked, this readily available data source can provide valuable information for localization, significantly reducing the reliance on costly site surveys.

Though trajectory data lacks explicit labels, relative positional information between RPs can still be derived from IMU readings. The key idea of WiMU is to exploit such information provided by WiFi RSSI fingerprints and IMU data collected from unlabeled RPs, and accordingly generate spatial representations for APs (See the lower half of Figure 1). These representations reside in a high-dimensional space, where spatial proximity in the physical world is preserved—i.e., physically closer APs are mapped to nearby points in the representation space. This raises a fundamental question: *how can we quantify AP proximity and derive their positions in this representation space?*

From user trajectories, we observe that inter-AP distances can be approximated by fusing WiFi and IMU data. Each trajectory is modeled as a graph, where nodes represent both RPs and APs. AP-to-AP distances are then inferred via shortest path lengths in the graph. Specifically, AP-to-RP distances are estimated using RSSI values and a signal propagation model [11], while RP-to-RP distances are derived from IMU data using the Pedestrian Dead Reckoning (PDR) algorithm. By combining these two estimates, approximate distances between APs can be obtained.

A natural follow-up question is how to generate spatial representations of APs from the estimated pairwise distances. Leveraging the graph structure inherent in WiFi networks, we model APs as nodes in a graph, with edge weights corresponding to the inferred inter-AP distances. To capture spatial dependencies, we employ a Variational Graph Auto-Encoder (VGAE) [12], which consists of a GNN-based encoder-decoder architecture. The encoder maps the graph structure to latent node representations, while the decoder reconstructs the graph from these embeddings. Through self-supervised training, the resulting AP embeddings effectively encode spatial proximity information. These embeddings are then aggregated using an RSSI-weighted scheme to generate representations for RPs. A regression model is then trained to map RP representations to physical coordinates. Once trained, both the AP embeddings and the regression model are reused for online localization: given a new RSSI fingerprint, the corresponding RP representation is computed and used to predict its location via the regression model.

However, due to the low sampling rate of WiFi signals, localization results are sparse in time. To bridge the gap between two consecutive WiFi readings, we incorporate a Pedestrian Dead Reckoning (PDR) algorithm based on IMU data for relative localization. Starting from the last predicted location, PDR estimates the user's position at each step until the next WiFi measurement is available. To enhance both stability and accuracy, a particle filter[13] is employed to fuse the localization results from IMU and WiFi data. Additionally, we design adaptive strategies to assess the confidence of each source and dynamically adjust their influence in the fusion process by tuning the filter parameters accordingly.

We implement WiMU and evaluate it on a large-scale Microsoft indoor location and navigation dataset [14]. We compare it with three state-of-the-art indoor localization approaches. Extensive experiments show that WiMU achieves an average localization error of 5.247 meters in commercial buildings ranging from less than 10,000 square meters to larger than 50,000 square meters, outperforming all the baselines by at least 32.8%. To further validate its practicality, WiMU is deployed in two real-world environments: a university campus and a logistics warehouse. In both settings, it successfully delivers accurate, real-time localization, demonstrating strong generalizability and robustness under diverse deployment conditions.

In general, we summarize our contributions as follows:

- We propose a practical site survey strategy that significantly reduces the effort required for collecting fingerprints in indoor localization.
- We present WiMU, an indoor localization system that leverages unlabeled multimodal sensor data to enable real-time and accurate localization.
- We evaluate WiMU on a large-scale public dataset and in real-world deployments, demonstrating its effectiveness and robustness across buildings of varying scales and layouts.

The rest of the paper is organized as follows. In Section 2, we review some related works. In Section 3, the system design of WiMU is presented in detail. Afterward, Section 4 shows the evaluation results. Eventually, we discuss and conclude our work in Section 6.

2 Related Work

Indoor localization. Indoor localization involves determining the position of humans or objects within indoor environments, forming the basis for numerous applications such as manufacturing, navigation, and healthcare [15]. Many studies leverage radio signals for indoor localization. For example, Ultra Wideband (UWB) technology is also favored due to its high bandwidth and low localization error [16]. Compared to UWB, Bluetooth Low Energy (BLE) offers lower power consumption and fewer privacy concerns, making it suitable for extensive deployment [17]. Additionally, some research explores using non-radio sensor data such as light [18], acoustic [19], and IMU signals [20]. Among all the technologies, WiFi is among the most popular for localization due to the wide deployment of WiFi, existing infrastructure, and easy accessibility [21, 22].

WiFi-based Indoor Localization. WiFi is among the most widely utilized wireless communication technologies. Typically, the reception range of WiFi signals is around 100 meters [23], making it suitable for indoor deployment. Therefore, researchers leverage WiFi physical information provided by abundant WiFi signal data for indoor localization purposes. WiFi signals can determine distances, angles, and travel times between APs and user devices. Additionally, Channel State Information (CSI) captures the scattering, fading, and power decay of WiFi communication channels, making it a valuable resource for positioning. With the different data modalities, techniques such as trilateration [24], Angle of Arrival [25], Time of Flight [26], and fingerprinting [6] can be employed individually or in combination for localization. Among these methods, fingerprinting is known for its precision and robustness, gaining considerable attention constantly.

WiFi fingerprint-based Indoor Localization. WiFi fingerprinting uses WiFi RSSI readings or CSI as unique identifiers for specific locations, akin to human fingerprints. Since collecting CSI data requires specialized hardware and software support, WiFi RSSI fingerprint is the more popular fingerprinting technology due to its simplicity and data accessibility [27]. Traditional fingerprinting involves manually collecting WiFi RSSI data at reference points (RPs) and annotating them with ground truth coordinates. The location is predicted by retrieving and comparing WiFi fingerprints from the database, then aggregating the coordinates of the most similar RPs [6, 28, 29]. With advancements in machine learning (ML), it has been shown that ML models can effectively extract patterns and withstand noise [30]. Various models have been applied to map original WiFi fingerprints to coordinates, effectively improving the localization accuracy [6–8]. However, most ML models require large amounts of labeled data for optimal performance and generalizability, which is a major limitation for the widespread deployment of ML-based WiFi fingerprinting technology.

Low-effort Indoor localization. To mitigate the labor-intensive nature of fingerprint-based indoor localization, research has increasingly focused on low-effort systems that automate map creation. One approach, exemplified by EZ[31], leverages crowdsourced Wi-Fi data to learn an environmental radio propagation model. By solving a complex joint optimization problem, this method bypasses the need for a fingerprint map altogether. However, its reliance on external GPS signals for location grounding is often impractical in confined or underground spaces. In contrast, Zee[32] fuses data from smartphone inertial sensors with map constraints to infer user trajectories. These automatically generated trajectories are then used to build a location-annotated Wi-Fi fingerprint database without explicit user input. A key limitation is its requirement for a detailed, pre-processed floorplan, which can introduce significant overhead in complex scenarios such as warehouses. A research thrust aims to eliminate reliance on any pre-existing infrastructure or maps. Systems based on Simultaneous Localization and Mapping (SLAM), such as 3D ActionSLAM[33], achieve this by concurrently building a map and tracking the user. A key innovation of 3D ActionSLAM is its use of recognized human activities as semantic landmarks to correct for sensor drift. Nevertheless, this approach requires a pre-trained human activity recognition model, and the effort of data collection, coupled with concerns about the model's generalizability, poses a notable challenge.

Graph Neural Network. Graph Neural Networks (GNNs) are a category of neural networks designed to operate on graphs. They efficiently process data represented as graphs, capturing the inherent patterns through a unique message-passing mechanism [34]. This mechanism enables nodes to exchange information iteratively, updating their representations through node aggregation. The resulting node representations encapsulate local features, facilitating various downstream tasks at the node, edge, and graph levels [35–37]. Given their capabilities, GNNs are widely used across diverse fields, including social networks, molecular biology, chemistry, and natural language processing [38–40]. In wireless communication scenarios, where networks can naturally be depicted as graphs, GNNs are also leveraged for network applications, such as resource allocation [41], wireless communication [42], and network routing [43].

Graph-based Localization. Several studies have explored this direction. For instance, MetaGraphLoc [44] constructs a graph where Access Points (APs) serve as nodes, and the edges are weighted by the signal similarity between them. It then utilizes a GNN to predict a device's precise coordinates by learning from a feature vector

that combines real-time RSSI readings and IMU data. Similarly, GraphLoc [45] employs a dual-graph strategy. It first models the building's physical layout and then creates a "Logical Floor Graph" by clustering WiFi fingerprints and their transitions. By matching these two graphs, it automatically generates a fingerprint map and uses Bayesian inference on RSSI scans to determine the user's current region. However, they both depend on massive, labor-intensive data collection and labeling. MetaGraphLoc, for example, required over 500,000 labeled samples to train and validate its model in a 9,000 m² building. Likewise, GraphLoc's deployment involved 25 participants over 33 days to collect over 170,000 labeled WiFi records across a 6,600 m² area. As another notable work, GraFin [46] explores the application of GNN to WiFi-based indoor localization. It defines both APs and labeled RPs as nodes in its graph structure, establishing connections between APs and RPs only when they are associated with the same WiFi scanning. This approach may lead to two severe problems: First, it's not scalable to use RPs as nodes since its number is highly related to collected data and any changes will affect the graph structure, leading to the re-training overhead. Secondly, GraFin's graph can be extremely sparse as all AP-AP and most AP-RP are disconnected. The two factors together undermine its scalability and effectiveness for indoor localization. More experimental results fully justify this point in the Evaluation section.

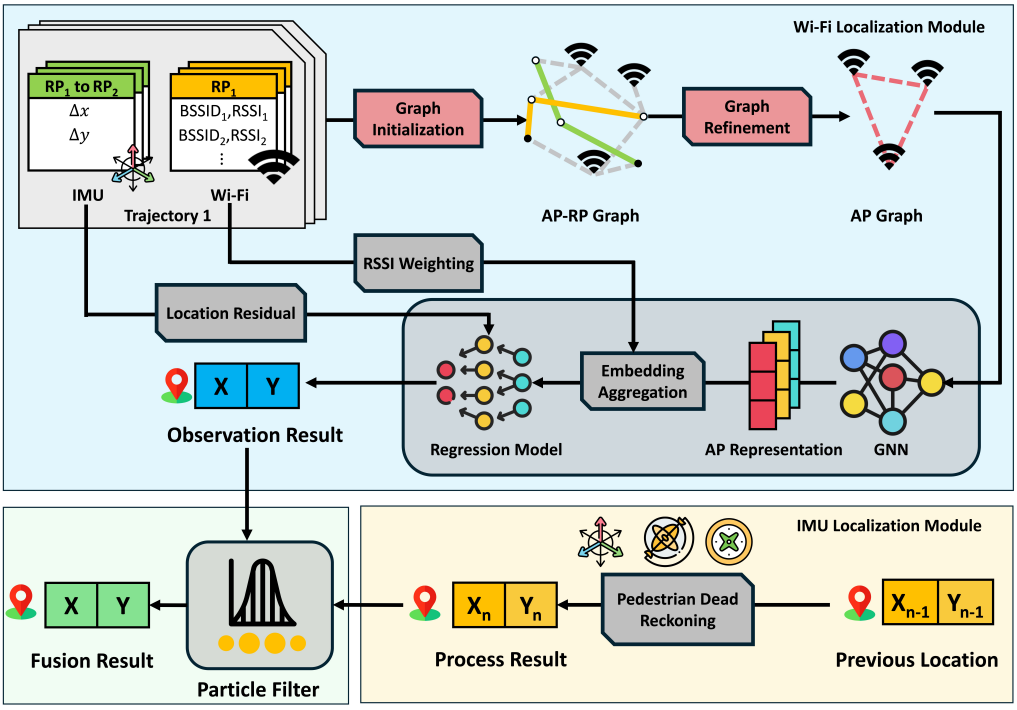


Fig. 2. System overview of WiMU.

3 System Design

3.1 Problem Definition

We consider a practical fingerprint collection setup in the real world where users move freely indoors. Their mobile devices, equipped with IMU and WiFi sensors, regularly collect data along their movement traces. The

WiFi RSSI fingerprint is defined as $fp = \{(BSSID_i, RSSI_i)\}_{i=1}^{k}$, where each pair comprises the BSSID (the MAC address of an Access Point) and its corresponding RSSI (Received Signal Strength Indicator) value. We denote a labeled RP as $r = \{fp, (x, y), t\}$ and an unlabeled RP as $r' = \{fp, t\}$. Here, (x, y) and t represent the location and timestamp, respectively. For each trace, a series of RPs $R = \{r'_1, r'_2, \ldots, r'_n\}$ is sampled, and most of them are unlabeled. Additionally, each trace also has corresponding IMU data, denoted as $I = \{(s, \theta)_t\}_{t=t_0}^{T}$. Each entry in I contains the step count s and the orientation angle θ at timestamp t, with s obtained from the step counter sensor and θ derived from the rotation vector sensor. Accordingly, the data collected from a trace can be represented as $l = \{R, I\}$, including both RP series and IMU data. All the collected data includes several traces, denoted as $L = \{l_1, l_2, \ldots, l_m\}$. The objective is to build a robust and high-accuracy indoor localization system with L and a few labeled RPs. It can accept WiFi RSSI fingerprints fp and IMU data I as input and outputs two-dimensional coordinates $(\hat{x}, \hat{y})$.

3.2 Overview

Figure 2 presents an overview of WiMU, encompassing both single-sensor localization and sensor fusion phases. In the single-sensor localization phase, WiMU independently estimates positions using WiFi and IMU data. For WiFi-based localization, WiMU first constructs an AP-RP graph from user trajectories. Pairwise distances between APs are estimated and used to refine the graph into an AP-only graph. A GNN model is then applied to learn AP embeddings, which are preserved for localization. An embedding aggregation module subsequently derives RP representations by aggregating nearby AP embeddings. These RP embeddings, along with their known coordinates, are used to train a regression model that maps representations to physical locations. Both the AP embeddings and the regression model are reused during online inference.

For IMU-based localization, WiMU extracts device orientation and stride information from IMU readings to compute displacement vectors. These are cumulatively added to the last known location to estimate the user's current position.

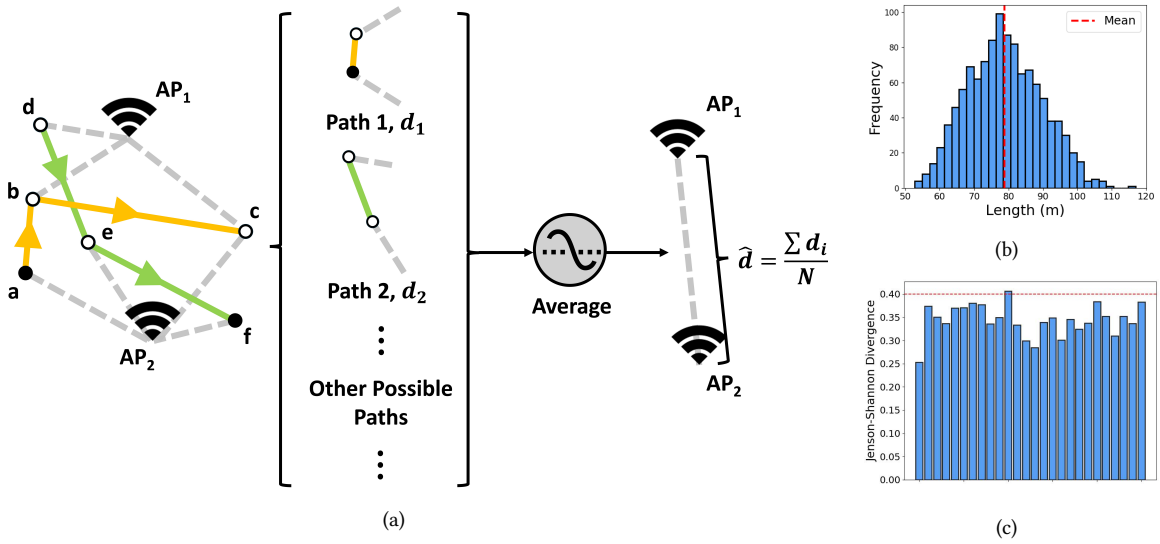
In the sensor fusion phase, predictions from both modules are integrated using a particle filter. The filter dynamically adjusts its noise parameters based on confidence indicators from sensor inputs, such as magnetometer and gyroscope readings. Higher noise reflects lower confidence in a prediction, thus reducing its influence on the final fused estimate.

3.3 WiFi Localization

3.3.1 Graph Initialization. To generate representations that reflect the spatial proximity of APs, we start by quantifying their proximity by measuring the pairwise distance between APs. However, determining the distances between APs is non-trivial because they cannot be directly obtained from WiFi or IMU data. To address this challenge, we introduce RPs as intermediates to connect APs since RP-AP and RP-RP are connectable. We propose to build an RP-AP graph first where the nodes consist of both APs and RPs, and the edges are defined as the distances between nodes. Figure 3(a) illustrates such an AP-RP graph. Nodes a to f represent RPs, where $a \rightarrow b \rightarrow c$ and $d \rightarrow e \rightarrow f$ refer to two traces. The graph also includes two AP nodes, AP_1 and AP_2, with dashed grey lines representing the connectivity to RPs. Based on the types of vertices, the edges in the graph are categorized into two types: AP-RP edges and RP-RP edges.

For the AP-RP edges, such as $d - AP_1$ or $e - AP_2$, their distance can be estimated with WiFi RSSI via a signal propagation model. Here, we use the Log-distance Path Loss (LDPL) model [11] to estimate the distance d^L_{i,AP_j} between an RP r_i and an AP AP_j, shown in Equation. 1:

$$d^L_{i,AP_j} = d_0 \cdot 10^{\frac{L-L_0}{10 \cdot n}} \tag{1}$$



(a) (b) (c)

Fig. 3. (a) The distance estimation between APs. (b) The path length distribution for one pair of APs. (c) Jensen-Shannon divergence for the path length distribution of 25 pairs of APs.

For the constant parameters, L_0 represents the signal strength received at a distance of d_0 meters from AP_j, and n is the path loss exponent related to the environment, usually set from 3 to 4 in indoor scenarios. For the variables, L is the signal strength received at r_i from AP_j. We use empirical values $d_0 = 1m, L_0 = -40dBm, n = 3$ [47] for simplicity.

Meanwhile, for the RP-RP edges like $a - b$ and $d - e$, their distances are estimated using IMU data through the PDR algorithm [48]. The PDR algorithm can estimate the coordinates of the next RP based on the locations of the previous RP, thereby calculating the distance between them. In the PDR algorithm, the process begins with step detection, where peaks of accelerometer signals are identified as steps. The stride length is then estimated by analyzing the magnitude of vertical acceleration, ranging from 0.4 to 0.8 meters dynamically [49]. Eventually, the heading orientation is derived from the rotation vector sensor by integrating the magnetometer readings from a built-in API[50]. Assume a user walks $k = s_{n+1} - s_n$ steps to reach the next RP from current RP r_n, the location $(x_{n+1}, y_{n+1})$of the next RP r_{n+1} is computed as follows:

$$x_{n+1} = x_n + \sum_{i=1}^{k} stride_i \cdot \cos\theta_i, y_{n+1} = y_n + \sum_{i=1}^{k} stride_i \cdot \sin\theta_i \tag{2}$$

where $stride_i$ is the stride length and θ_i is the angle of the heading orientation of each step. After obtaining the coordinates, the distance between two RPs can be calculated with their coordinates:

$$d^P_{n,n+1} = \sqrt{(\sum_{i=1}^{k} stride_i \cdot \cos\theta_i)^2 + (\sum_{i=1}^{k} stride_i \cdot \sin\theta_i)^2} \tag{3}$$

Through this process, we have defined an AP-RP graph and computed its edge weights. It's worth noting that we don't directly use the AP-RP graph to generate representations because the number and locations of RPs are closely related to the data collection process, making the graph dynamic. As the number of RPs grows, the size

of the AP-RP graph may become very large. In contrast, the number and locations of APs are relatively stable, resulting in an AP graph with a stable size and fixed nodes.

3.3.2 Graph Refinement. In this subsection, we utilize the AP-RP graph to obtain distance relationships between APs and refine it to an AP graph, where each node refers to an AP and each edge represents the distance between two APs.

To address this challenge that AP-to-AP distances are unavailable from existing sensor data, we propose to build the path between APs based on the AP-RP graph. For example, shown as in Figure 3(a), the path 1 denoted as $AP_1 - b - a - AP_2$ actually connect two APs and its distance d_1 can be calculated as follows:

$$d_1 = d^L_{b,AP_1} + d^P_{a,b} + d^L_{a,AP_2} \tag{4}$$

Here, d^L_{b,AP_1} and d^L_{a,AP_2} represents the distances between corresponding AP and RP. The $d^P_{a,b}$ represents the distances between RP a and RP b. However, as shown in Figure 3(a), we can observe that there are many possible paths between two APs in addition to path 1, such as $AP_1 - d - e - AP_2$, $AP_1 - c - AP_2$, etc. We define the lengths of all possible paths between AP1 and AP2 as a set $D_{1,2} = \{d_1, d_2, \cdot, d_N\}$. Since different lengths in D refer to different paths, which one is the closest to the actual distances between AP1 and AP2 remains a problem.

To this end, we then conduct an experiment to investigate the distribution of $D_{1,2}$. As shown in Figure 3(b), we observe that D exhibits an approximately normal distribution. Subsequently, we use the Jensen-Shannon divergence (JSD) [51] to support our observation by quantifying the similarity between such distribution and normal distribution. A smaller JSD indicates that the two distributions are more similar to each other. In Figure 3(c), we demonstrate the JSD of D for 25 random pairs of APs on one floor. The JSD generally stabilizes between 0.3 and 0.4. Besides, we sample 30 buildings and calculate the average JSD for all the AP pairs, resulting in 0.323. As shown by the red line in Figure 3(c), the distribution can be recognized as normal distribution-like when JSD is less than 0.4 [52, 53]. Therefore, we assume that the path length distribution of the APs follows a normal distribution and use the mean value to estimate the actual distance between the APs:

$$\hat{d} = \frac{1}{N}\sum_{i=1}^{N} d_i, d_i \in D = \{d_1, d_2, \cdots, d_N\} \tag{5}$$

With the distance estimations between APs, we refine the AP-RP graph to an AP graph. This AP graph only consists of AP nodes and can be formulated as $\hat{A}$:

$$\hat{A}_{ij} = \begin{cases} \hat{d}_{ij}, \text{if there is a path between AP } i \text{ and AP } j \\ \infty, \text{otherwise} \end{cases} \tag{6}$$

where $\hat{d}_{ij}$ refers to the estimated distance between AP i and AP j.

3.3.3 Spatial Representation Learning. With the refined AP graph, how to utilize it for representation generation remains a challenge. We notice that Graph Neural Networks (GNN) has demonstrated an impressive ability to capture the information embedded in graphs [34], which have been widely used across various domains [38–41]. Therefore, we introduce GNN to learn AP representations from the AP graph.

Before applying a GNN on the AP graph $\hat{A}$ to generate representations for AP, we need to transform it to meet the requirements of the GNN. In GNN, the node representations are obtained by aggregating the representations of neighboring nodes, indicating that the neighboring representations with higher edge weights have greater impacts on the node's representation. This means the graph should reflect the similarity of nodes. The closer the APs are, the greater the edge weight should be. However, the current AP graph reflects the dissimilarity between

APs by distances. Therefore, we need to transform the AP graph $\hat{A}$ to another similarity graph A:

$$A_{ij} = \begin{cases} 1 + tanh(-10 \cdot \delta(\hat{A}_{ij})), \hat{A}_{ij} \neq \infty \\ 0, \text{otherwise} \end{cases} \tag{7}$$

In Equation. 7, we first use Min-Max function δ to normalize the original AP graph A, then employ a monotonically decreasing function $f(x) = 1 + tanh(-10 \cdot x)$ to convert the normalized distances into similarities. This function not only ensures that larger input distances result in lower output similarities but also amplifies the impact of distances on similarities.

With the similarity graph A, we apply a variational graph auto-encoder (VGAE) [12] model to generate AP representations. VGAE is a framework for self-supervised learning on graph-structured data based on variational auto-encoder [54]. It's capable of learning interpretable and efficient representations for undirected graphs. In our cases, it's aimed to generate node representations to characterize APs' positional features by capturing the relative location relations from the AP graph. The main structure of VGAE consists of an encoder and a decoder. The encoder comprises consecutive graph convolution layers, and the decoder is an inner product decoder with a sigmoid activation function. During the training process, the VGAE model first randomly initializes a representation matrix $X_0^{N \times m}$, where each row represents the initial representation of an AP. This matrix X_0 is then fed into the encoder. Each convolution layer in the encoder updates the node representation by aggregating the representations of neighboring nodes:

$$X_{n+1} = \phi(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X_n W_n) \tag{8}$$

where X_n is the input representation matrix of the n-th convolution layer. $\tilde{A} = A + I$ is the similarity graph with self-loop, while $\tilde{D}$ is the degree matrix of $\tilde{A}$. ϕ and W_n represent the activation function and weight, respectively. The output of the last convolution layer is the representations of APs, denoted as $Z^{N \times d}$. Subsequently, Z is then input to the decoder to reconstruct the similarity graph $A^{'}$ by applying an activation function ϕ (i.e. sigmoid) after the inner product operation on Z:

$$A^{'} = \phi(ZZ^T) \tag{9}$$

Next, the reconstructed graph $A^{'}$ is compared with the original graph A to compute the loss and update the model. Further details are provided in Section. 3.3.5.

3.3.4 Localization with Spatial Representation. The final stage of the offline training phase involves mapping learned representations to physical coordinates. Since the locations of APs are unknown, we compute representations for labeled reference points (RPs) using the existing AP embeddings. These RP representations, paired with their known coordinates, serve as training data for a regression model. Specifically, for a labeled RP $r = \{fp, (x, y), t\}$, its representation $\mathbf{e}$ is computed by aggregating the embeddings of associated APs, weighted by their corresponding RSSI values:

$$\begin{aligned} w_i &= \frac{1}{1 + d(RSSI_i)} \\ \mathbf{e} &= \sum_{i=1}^{n} \left\{ \frac{w_i}{\sum_{j=1}^{m} w_j} \mathbf{z}_i \right\}, (BSSID_i, RSSI_i) \in fp \end{aligned} \tag{10}$$

Here, d is the LDPL model to convert the RSSI to distance, and w_i is the weight of AP i. The $\mathbf{z}_i$ refers to the representation of AP i. Afterward, a regression model is applied to localization, which takes the RP representation as input and outputs predicted locations:

$$(\hat{x}, \hat{y}) = \sigma(W_r \cdot \mathbf{e}) \tag{11}$$

where W_r denotes the parameters of the regression model, and σ represents the sigmoid function used to project the model's output into the 0,1 range. We adopt a multi-layer perceptron (MLP) as the regression model to learn the mapping from RP representations to physical coordinates. The detailed training procedures are described in Section 3.3.5 and Section 3.3.6.

3.3.5 Pre-training. To facilitate model convergence, we leverage unlabeled data by introducing pre-training tasks. The first task is graph reconstruction, a widely adopted objective in representation learning. The associated loss function comprises two components: a reconstruction loss and a Kullback–Leibler (KL) divergence term [12]. The reconstruction loss quantifies the similarity between the original graph and the reconstructed graph, encouraging structural consistency. The KL divergence regularizes the latent representations by aligning them with a prior distribution (typically a standard normal distribution), thereby mitigating overfitting [12, 55]. The complete loss function is defined as follows:

$$\begin{aligned} \mathcal{L}_{recon} &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}||A_{ij} - A_{ij}^{'}||^2 \\ \mathcal{L}_{KL} &= \frac{1}{2}\frac{1}{Nd}\sum_{i=1}^{N}\sum_{j=1}^{d}(\mu_{ij}^2 + \sigma_{ij}^2 - 2log(\sigma_{ij}) - 1) \\ \mathcal{L}_1 &= \mathcal{L}_{recon} + \mathcal{L}_{KL} \end{aligned} \tag{12}$$

where μ and σ are the mean and the standard deviation of Z generated by two individual convolution layers applied sequentially after the encoder. By minimizing the loss $\mathcal{L}$, the encoder is optimized to fuse the spatial information into representations. More importantly, the whole training process of the VGAE model is in a self-supervised way and does not require any labeled RP data.

The second pre-training task is displacement prediction. In this task, the model estimates the locations of consecutive unlabeled RPs and computes the displacement between them. These predicted displacements are then compared against the ground-truth displacements $d_{n,n+1}^{P}$, as defined in Eq. 3. This task enables joint training of both the regression model and the GNN, encouraging the learned representations to preserve spatial consistency. The corresponding loss function is formulated as:

$$\mathcal{L}_2 = \frac{1}{N}\sum_{i=1}^{N}||d_{i,i+1}^{P} - \sqrt{(\hat{x_i} - \hat{x_{i+1}})^2 + (\hat{y_i} - \hat{y_{i+1}})^2}|| \tag{13}$$

It is important to note that only pairs of consecutive unlabeled RPs from the same trajectory are used in this task, as the displacements $d_{n,n+1}^{P}$ inferred from IMU data are relatively accurate over short intervals, mitigating the influence of cumulative IMU errors on model training. Finally, the total pre-training loss is computed by combining the graph reconstruction loss $\mathcal{L}_1$ and the displacement prediction loss $\mathcal{L}_2$ using a weighting factor α to balance their contributions. Based on empirical observations, α is set to 10:

$$\mathcal{L}_{pre} = \mathcal{L}_1 + \alpha \cdot \mathcal{L}_2 \tag{14}$$

The pre-training phase is conducted in an unsupervised manner, relying solely on unlabeled trajectory data without the use of any location annotations. After pre-training, the parameters of the GNN are frozen to retain the graph-structured knowledge learned from the data. During the subsequent fine-tuning phase, only the parameters of the regression model are updated, allowing it to adaptively improve the mapping from learned representations to physical coordinates.

3.3.6 Fine-tuning. After pre-training, the model effectively captures the relative positional relationships among RPs. To adapt the model to the target localization domain, it is fine-tuned using a small set of labeled data. Given that the pre-trained model already provides reliable estimates of inter-RP distances, this spatial calibration requires only a limited number of labeled samples. Let the labeled RP dataset be denoted as $R = \{r_1, r_2, \cdots, r_M\}$, and their corresponding representations as $R_e = \{e_1, e_2, \cdots, e_M\}$, computed according to Eq. 10. The fine-tuning loss function is defined as follows:

$$\mathcal{L}_{fine} = \frac{1}{M} \sum_{i=1}^{M} ||\sigma(W_r \cdot e_i) - (x_i, y_i)|| \tag{15}$$

The fine-tuning stage is only used to update the regression model. Subsequently, the GNN and the regression model remain frozen for online inference.

3.4 IMU Localization

To address the sparse temporal resolution of WiFi-based localization, IMU data is utilized to interpolate user positions between consecutive WiFi samples, leveraging its higher sampling rate. The Pedestrian Dead Reckoning (PDR) algorithm is employed to estimate step-wise displacements in real time. However, not all smartphones are equipped with built-in step detectors. To ensure compatibility, we implement a custom step detection algorithm based on accelerometer data, consisting of four components: peak detection, rise detection, peak-to-valley verification, and interval verification.

First, a peak candidate is identified when the acceleration exceeds a threshold th_p. To reflect human gait patterns, only two consecutive accelerations are considered valid for step detection. Next, a peak-to-valley difference threshold th_d is applied to eliminate insignificant fluctuations. Finally, due to physiological constraints, users are unlikely to take more than one step within 250 milliseconds; thus, candidates within this interval from the previous step are discarded. Only peaks that pass all verification steps are recognized as valid steps.

To improve robustness in dynamic environments, both th_p and th_d are updated adaptively based on the average of the four most recent peak accelerations and peak-to-valley differences, respectively. Once a step is detected, heading orientation is obtained from the device's rotation vector sensor using a standard API [50]. The location of each step is then calculated using Eq. 2, with the displacement applied from the last known position.

3.5 Online Inference

After obtaining localization results from both WiFi and IMU data, a natural question arises: *which result should be more trusted?* Drawing on Bayesian filtering principles, we adopt a particle filter to fuse the two sources for more accurate and stable position estimation. The particle filter naturally balances these two: it smooths short-term motion using PDR while periodically anchoring to absolute positions from WiFi. This design leverages the strengths of both sources while limiting their weaknesses, particularly the drift associated with IMU. In this context, the IMU-based localization module serves as the prediction model, using system inputs (i.e., IMU readings) and the previous position to estimate the current location. In contrast, the WiFi-based module acts as the observation model, directly producing location estimates from environmental measurements.

The fusion process begins by initializing a particle cloud centered on the initial WiFi localization estimate, with added Gaussian noise. Each particle represents a hypothetical position and is assigned an equal initial weight. In the prediction step, particles are propagated forward using the IMU-based displacement model, simulating potential user movements. Then, in the update step, particle weights are adjusted according to the likelihood of the current WiFi observation, with higher weights assigned to particles closer to the observed WiFi-based location. This process allows the filter to balance short-term motion continuity with more reliable but sparse WiFi observations.

The overall particle filtering procedure is formalized as follows:

$$\begin{aligned} x_i &= x_i + \delta x + \epsilon_{x_i} \\ y_i &= y_i + \delta y + \epsilon_{y_i}, \epsilon_{x_i}, \epsilon_{y_i} \sim \mathcal{N}(0, \sigma_{sys}^2) \\ w_i &= e^{-\frac{(x_i - x_{obs})^2 + (y_i - y_{obs})^2}{2\sigma_{obs}^2}} \end{aligned} \tag{16}$$

Here, (x_i, y_i) represents the i-th particle, $(\delta x, \delta y)$ denotes the displacement derived from the IMU readings, and (x_{obs}, y_{obs}) is the observed location inferred from the WiFi data. The influence of each localization module is governed by its estimated uncertainty, represented by noise parameters σ_{sys} and σ_{obs}. A higher σ value signifies lower confidence in a module's output, thereby reducing its weight in the joint position estimation. To account for real-world dynamics, we propose a rule-based method for adaptively tuning these noise parameters (σ_{sys} for the IMU, σ_{sys} for WiFi). This is critical because the IMU module's predictions can be degraded by factors such as inconsistent user posture and electromagnetic interference. Similarly, the WiFi localization module's reliability diminishes in areas with weak average RSSI. By dynamically quantifying the uncertainty of each module, our system effectively mitigates the impact of transient noise and lead to a more stable and robust localization outcome.

The system noise σ_{sys} is dynamically adjusted based on the local magnetic field strength and the orientation of the device, as both factors influence the accuracy of heading estimation and thus the displacement inference. The observation noise σ_{obs} is determined by the average RSSI value of the detected WiFi access points. When the average RSSI falls below -65 dBm, it is assumed that the majority of AP signals are weak, resulting in less reliable WiFi fingerprints and degraded localization accuracy. The principles for adjusting these two hyperparameters are summarized in Table 1. These values are empirical parameters, initially determined through laboratory testing. Their effectiveness was subsequently validated in diverse real-world scenarios, including the on-campus and warehouse case studies presented in Section 5.

Table 1. Adjustment Rules for σ_{sys}^2 and σ_{obs}^2

Parameter	Condition	Value
σ_{sys}	Valid posture **and** valid magnetic field	2.0
	Only one (either posture or magnetic field) is valid	3.0
	Neither is valid	4.0
σ_{obs}	Average RSSI > −65 dBm	1.5
	Average RSSI ≤ −65 dBm	2.5

The device posture is considered valid if the phone's pitch lies within $[-60°, +60°]$ and roll within $[-30°, +30°]$. The magnetic field is deemed reliable when its measured strength remains below $80\mu T$. These conditions ensure accurate IMU-based heading estimation, thus contributing to the dynamic adjustment of the system noise parameter. Lastly, the particles are aggregated based on their weights to generate the fused localization result. To prevent a few particles from dominating the weight distribution, multinomial resampling is performed—selecting new particles proportionally [56].

4 Evaluation

4.1 Experiment Setup

To evaluate the performance of WiMU, we implement WiMU with PyTorch [57] and compare it with other baseline models on a public dataset.

4.1.1 Dataset. We evaluate WiMU using the dataset from the Microsoft indoor location and navigation competition [14], which consists of dense indoor signatures of WiFi, IMU sensors, iBeacons, and more, along with ground truth coordinates collected from 204 buildings and 982 floors in various cities across China. The data is organized as a sampling trace walked by a site surveyor. We randomly select 10 floors with sizes less than 10,000 square meters, 10 floors between 10,000-50,000 square meters, and 10 floors more than 50,000 square meters. These floors are categorized into small, medium, and large tiers according to their sizes. Based on the size of the floors, the number of RPs varies from tens to thousands, with only approximately 10-20% being manually labeled. We randomly select labeled RPs and performed data augmentation. The augmented RPs, along with the selected RPs, are used as the training set for the regression model. The remaining labeled RPs are then used as the test set to evaluate the performance.

4.1.2 Baseline models. We choose 3 baselines based on WiFi RSSI fingerprinting, including GraFin [46], WePos [58], and MYRCJ [59]. GraFin and WePos are both machine learning systems, while MYRCJ is a traditional signal processing approach. The following are the details of these baseline models:

■**GraFin** is a GNN-based localization system. It focuses on generating representations and performing localization based on them. GraFin constructs a graph consisting of both APs and labeled RPs and trains a graph convolution network (GCN) to generate representations. Additionally, it employs a k-nearest neighbor (KNN) method to match the representations and aggregate retrieved coordinates to predict the location. As we mentioned, it suffers from graph sparsity and rapid performance degradation when the number of labeled RPs is limited.

■**WePos** is a weakly supervised localization system based on BERT. It formulates the WiFi RSSI fingerprint as sentences, with each BSSID-RSSI pair treated as a token. WePos is pre-trained by performing the next sentence prediction (NSP) task to predict the subsequent RP given a previous RP. Subsequently, WePos utilizes the generated representations in a downstream classification task to achieve merchant-level localization. Since it also used an unsupervised training method to generate representations according to spatial relationships, we adapt it to perform the absolute localization task.

■**MYRCJ** is the second-place solution in the Microsoft indoor location and navigation competition. Comparing with the winner solution TRACK-ME-IF-YOU-CAN[59] which used a lot of manual efforts on data preprocessing, we believe MYRCJ is a more suitable baseline. It initially utilizes the PDR algorithm to predict the relative positions of RPs. Consequently, a traditional fingerprint method is utilized to determine the absolute locations calibrated by relative positions. Finally, it employs a post-processing technique called snap-to-grid to improve localization accuracy. The hyperparameters of MYRCJ are carefully set through site surveys.

4.1.3 Metric. We compare the performances of WiMU and baseline models with mean absolute localization error (MAE), which is defined as $\sum_{i=1}^{n} \frac{\sqrt{(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2}}{n}$, where (x_i, y_i) and $(\hat{x}_i, \hat{y}_i)$ are the ground truth coordinates and predicted coordinates of i-th labeled RP, respectively.

4.1.4 Hyperparameter. We have configured WiMU with the following settings. For the LDPL model, we empirically set $L0 = -40$ dBm and $n = 3$ [47]. For the VGAE model, we define the input and output dimensions as 116, with hidden dimensions of 512. The MLP model uses four linear layers with output dimensions of 256, 128, 64, and 2. Leaky ReLU is the activation function for the first three layers while the last layer uses a sigmoid activation function. The models are optimized by Adam optimizer with an initial learning rate $l = 0.001$ and trained WiMU for 300 epochs.

4.2 Overall Performance

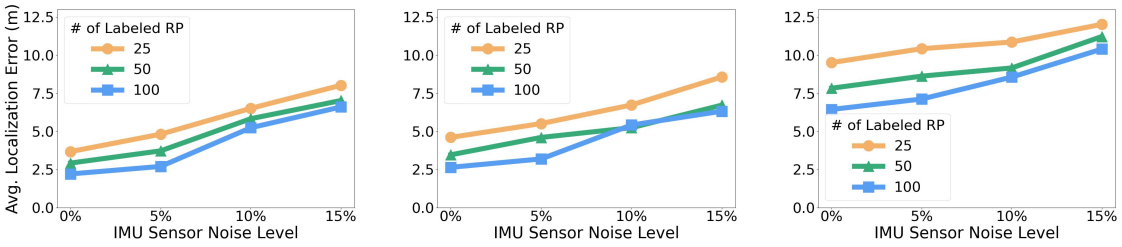
Table 2 summarizes the performance of WiMU alongside baseline models. Each cell presents the average absolute localization error (in meters) across varying building sizes and numbers of labeled RPs. The average localization errors for the four models are 5.247, 7.813, 13.922, and 36.825 meters, respectively. In large buildings, when the

Table 2. Mean Absolute Error (MAE) in meters of WiMU and baseline models across different settings. Percentages indicate the increase in MAE compared to the Small building size.

Model	# of RP	Small (Baseline MAE)	Medium (MAE / Change)	Large (MAE / Change)
WiMU	25	4.910	5.293 (+7.8%)	9.209 (+87.6%)
	50	3.851	3.875 (+0.6%)	7.556 (+96.2%)
	100	3.016	3.158 (+4.7%)	6.353 (+110.6%)
	Avg		5.247 (overall average)	
GraFin	25	7.418	8.075 (+8.9%)	15.828 (+113.4%)
	50	5.824	5.901 (+1.3%)	9.876 (+69.6%)
	100	4.955	4.448 (-10.2%)	7.993 (+61.3%)
	Avg		7.813 (overall average)	
MYRCJ	25	10.622	13.324 (+25.4%)	37.366 (+251.8%)
	50	4.875	8.657 (+77.6%)	26.136 (+436.1%)
	100	3.517	4.387 (+24.7%)	16.410 (+366.6%)
	Avg		13.922 (overall average)	
WePos	25	16.929	30.352 (+79.3%)	70.953 (+319.1%)
	50	15.172	29.294 (+93.1%)	65.916 (+334.5%)
	100	14.520	25.539 (+75.9%)	62.747 (+332.1%)
	Avg		36.825 (overall average)	

number of labeled RPs decreases from 100 to 25, the accuracy of the models declines by 2.295, 4.642, 12.333, and 5.143 meters, respectively. MYRCJ, which relies on raw RSSI fingerprints, is impacted by fluctuations and noise in the original WiFi signals. Similarly, WePos encounters accuracy loss as it cannot process continuous RSSI values effectively, treating them as discrete values instead. GraFin experiences significant performance degradation due to the sparsity of its graph when the number of labeled RPs is limited, resulting in many isolated nodes. For these isolated nodes, the GNN model fails to produce effective representations. Among the 30 graphs generated by GraFin, 12.6% to 39.7% of the nodes are found to be isolated.

In contrast, WiMU outperforms all baseline models, demonstrating robust and consistent performance. It achieves the lowest localization error across all experimental settings. Specifically, with 100 labeled RPs, WiMU attains the lowest average localization errors of 3.016, 3.158, and 6.353 meters on three differently sized floors, representing at least a 14.2% reduction in error compared to the baselines. Furthermore, as floor size increases from small to large, the average localization error for WiMU rises by only 3.780 meters, whereas the errors for the



(a) Small (b) Medium (c) Large

Fig. 4. The average localization errors of WiMU with different IMU sensor error levels.

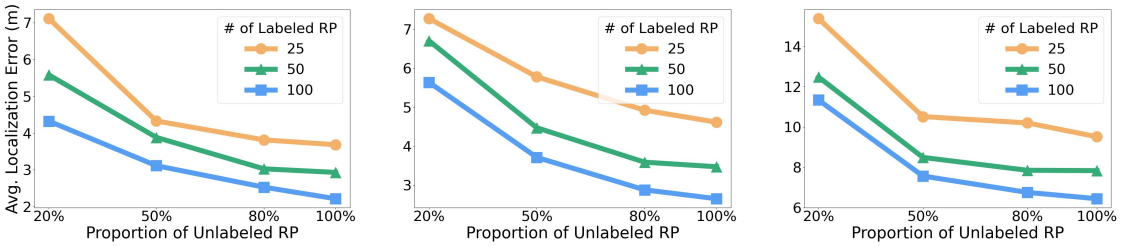
other baseline models increase by at least 5.167 meters. This highlights WiMU's resilience in handling large and complex building environments. Unlike the baseline models, WiMU does not depend on labeled RPs to construct its graph, and when the number of labeled RPs decreases from 100 to 25, its performance degrades by only 2.295 meters. In summary, WiMU achieves the highest accuracy while maintaining robustness and scalability, showing minimal sensitivity to variations in building size and the quantity of labeled data.

We also investigated the reasons for the decline in localization accuracy as floor size increases, identifying three primary factors. First, smaller floors in our dataset, such as food courts, tend to have simpler layouts. In contrast, larger floors are often complex retail environments, which present a more challenging mapping problem. Second, we observed a strong inverse correlation between floor size and AP density. On average, small-sized floors feature one AP per 26 m^2. This density decreases significantly for medium-sized floors (one AP per 76 m^2) and large-sized floors (one AP per 103 m^2). Such AP sparsity creates a fundamental challenge: it makes the Wi-Fi fingerprints of nearby locations less distinguishable. When multiple physical positions map to similar or even identical signal patterns, the model's ability to resolve location is severely compromised, directly causing the observed performance degradation. Finally, with a fixed number of unlabeled RPs, the data becomes too sparse on an expansive floor to effectively train a robust model. These factors collectively explain the observed performance degradation in larger environments.

4.3 Impact of Noisy IMU Sensors

WiMU extensively uses IMU sensors during both training and inference, meaning that noisy IMU readings can affect the robustness and effectiveness of the system. Although our dataset was collected with commercial smartphones that have inherent sensor errors, we conducted a simulation to evaluate the system's performance under varying noise levels. This experiment mimics scenarios involving factors like inaccurate sensors in low-cost phones or electromagnetic interference. We tested four noise levels: 0%, 5%, 10%, and 15%, which can be considered as common noisy levels from mobile devices. For each azimuth and distance value derived from the sensor data, we added a random error drawn uniformly from a range up to the specified noise level (e.g., up to 15% for the 15% noise level).

Figure 4 shows that while localization error is positively correlated with IMU noise, where WiMU exhibits graceful degradation. This resilience is rooted in the system's core design. During the offline training phase, WiMU builds its AP graph based on the relative proximity of RPs rather than their precise metric distances, making it inherently tolerant to estimation noise. This tolerance is further enhanced by the crowdsourcing approach, which statistically mitigates random IMU errors by aggregating numerous trajectories. Consequently, the WiFi localization module remains robust against IMU noises. During the online inference phase, a particle filter dynamically corrects the drift-prone PDR by re-weighting particles based on their alignment with the more



(a) Small (b) Medium (c) Large

Fig. 5. The average localization errors of WiMU with different proportions of unlabeled RPs.

accurate WiFi location estimates. In summary, WiMU is both theoretically and practically robust against typical levels of IMU noise.

4.4 Impact of Signal Propagation Model

Our system utilizes the Log-Distance Path Loss (LDPL) model for distance estimation. To assess the system's architectural flexibility and its compatibility with different signal models, we performed an ablation study by substituting LDPL with the Free-space Path Loss (FSPL) model. The FSPL model assumes an idealized, unobstructed line-of-sight path, which serves as a generalized baseline. It is formulated as follows:

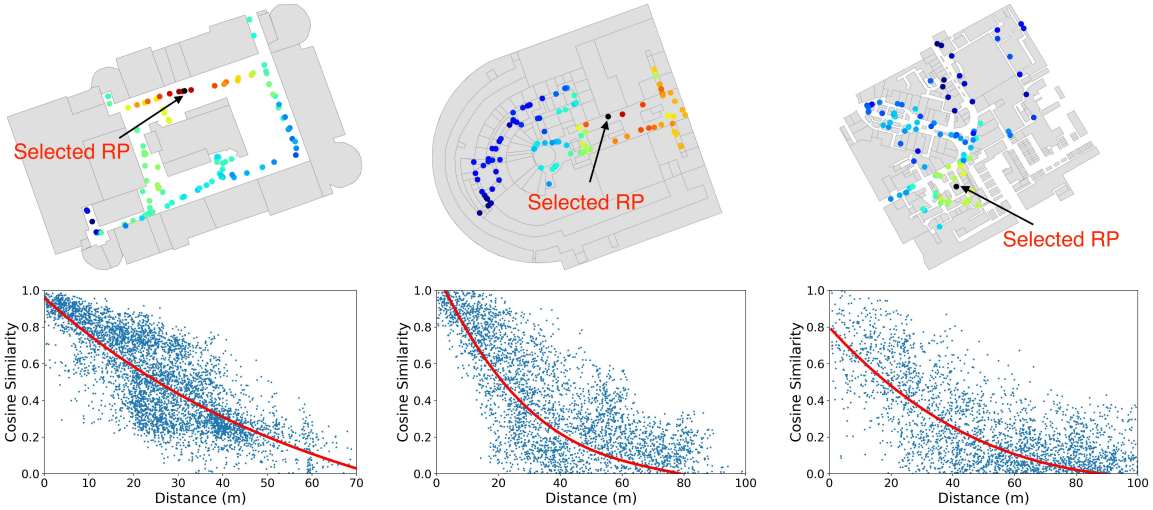
$$L_{\text{FSPL}}\ (\text{dB}) = 20\log_{10}(d) + 20\log_{10}(f) + 20\log\frac{4\pi}{c} \tag{17}$$

where $L_{\text{FSPL}}$is the path loss, d is the distance, f is the signal frequency, and C is a constant. In our implementation, the path loss is calculated as the difference between the AP's transmission power (assumed to be a constant 20 dBm) and the measured RSSI. Since the frequency f is available in the WiFi scan data, the distance d can be derived by inverting Equation 17.

Figure 7 illustrates the comparative results. The key insight is that the performance of WiMU is not contingent on a specific signal model. Across environments ranging from small to large-scale, the localization accuracy of the LDPL and FSPL models remains highly comparable. This result suggests that the architectural design of WiMU is robust and capable of delivering reliable performance with different signal propagation models and allowing for flexible model selection.

4.5 Impact of Unlabeled RP

We claim that even though most RPs are unlabeled, they can still provide spatial information to help generate representations. Therefore, we adjust the number of unlabeled RPs to study its impact on system performance. Figure. 5 illustrates the results, with each sub-figure representing a different floor size and each line indicating performance with varying numbers of labeled RPs. Since the number of available unlabeled RPs varies widely from one floor to another. Therefore, to ensure a normalized and consistent evaluation, we opted to use a fixed proportion of this data for training, as opposed to a fixed count. The x-axis represents the proportion of unlabeled RPs used in graph construction, while the y-axis represents the average localization errors. As the proportion increases, the model performance improves because more unlabeled data provide more distance estimations, making the estimated AP distances more accurate. Based on more precise distances, we can construct higher-quality graphs and generate better representations.



(a) Small (b) Medium (c) Large

Fig. 6. Top: Visualization of the similarity between RP representations. Bottom: The relationship between RP pairwise distances and the cosine similarity of representations.

4.6 Visualization of Spatial Representation

To evaluate the effectiveness of learned spatial representations, i.e., closer distance should lead to higher representation similarity, we devise a visualization approach. Since the ground truth locations of APs are unknown, it's difficult to evaluate the relationship between AP representations and AP pairwise distances. Therefore, we use the representations of labeled RPs to indirectly verify the effectiveness of the learned representations. Figure 6 visualized the similarity between RP representations among three different-sized floors. The top images show the cosine similarity of the representations between a randomly selected RP and other RPs. Warmer colors indicate higher similarity whereas cooler colors indicate lower similarity. These images align with our assumption that closer RP should have similar representations. Additionally, the bottom images illustrate the relationship between the pairwise distances and the representation similarities of all RP-RP pairs on the same floor. The red line is a cubic fitting curve obtained from the blue data points. It indicates an inverse correlation between distance and similarity, further confirming the ability of RP representations to reflect spatial information.

4.7 Impact of Graph Construction

To validate the effectiveness of our AP graph construction, we benchmark our distance-based method in WiMU against two baselines: a count-based and a Pearson-based approach. Figure 8 shows the average localization error for each method across different floor sizes, where the main bars represent the performance with 50 labeled RPs and the error bars indicate the results for 25 RPs (upper bound) and 100 RPs (lower bound). The results clearly demonstrate the superiority of our distance-based approach across all scenarios. On the small floor, our method achieves an average error of approximately 2.9 m. This represents a significant 34% reduction in error compared to the count-based method (4.4 m) and a 7% reduction against the Pearson-based method (3.1 m). The advantage becomes even more pronounced on the large floor, highlighting our method's scalability. While WiMU maintains a robust error of 7.9 m, the Pearson-based method's performance degrades severely to 14.2 m. In this

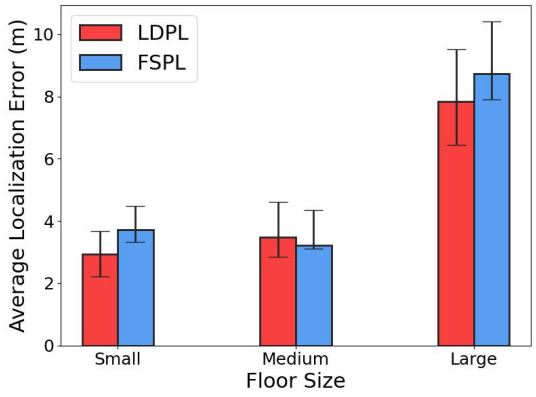


Fig. 7. Impact of signal propagation model.

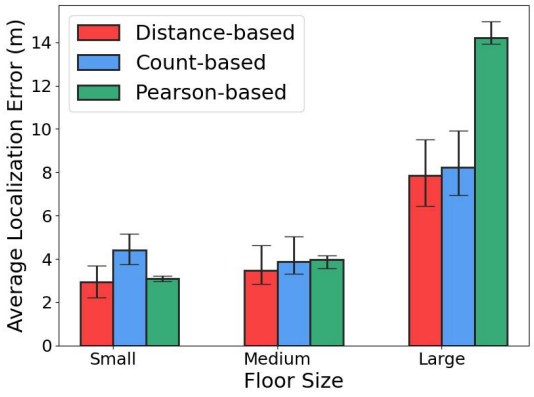


Fig. 8. Impact of graph construction.

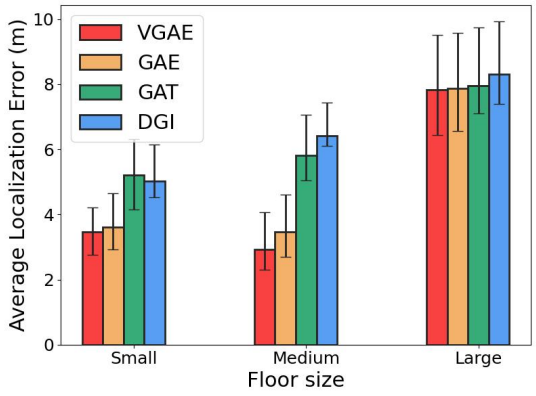


Fig. 9. Impact of GNN model.

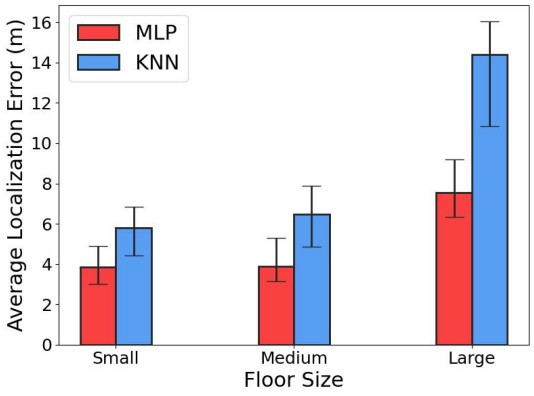


Fig. 10. Impact of regression model.

scenario, our method outperforms the count-based (8.3 m) and Pearson-based methods by approximately 5% and a substantial 44%, respectively. These quantified improvements confirm that by integrating RSSI and IMU data, our distance-based approach constructs a higher-quality graph that better reflects the true spatial proximity of APs and is more robust in larger, more complex environments.

4.8 Impact of the GNN model

We devise an ablation study to evaluate the impact of various GNN models. We choose other three GNNs, including graph auto-encoder (GAE) [12], graph attention network (GAT) [60], and DeepGraphInfomax (DGI) [61] to replace the VGAE model. GAE is another auto-encoder without distribution constraints. GAT introduces the attention mechanism, enabling the neural network to capture complex node relationships. DeepGraphInfomax leverages mutual information maximization between local and global graph representations to capture rich structural information. As shown in Figure 9, VGAE achieves the best performances among all GNN models, while other models also have satisfactory localization accuracy. This result confirms that WiMU is a model-agnostic method, allowing flexible switching of models according to different scenarios.

4.9 Impact of the Regression model

The MLP model is trained to map RP representations to coordinates with limited labeled data. To demonstrate the effectiveness of the MLP model, we replaced it with a non-parametric K-nearest neighbor (KNN) model and evaluated the localization accuracy. For each RP representation, the KNN model retrieves the nearest labeled RP representations based on cosine similarity and predicts coordinates by weighted summation of their coordinates [3]. Since GraFin uses the KNN model to predict locations, we adapt its weight calculation method here [46].

Figure 10 shows the evaluation results. Each bar represents the average localization error with 50 labeled RPs for small, medium, and large floors. The upper end of each error bar represents the average error with 25 labeled RPs, while the lower end represents the average error with 100 labeled RPs. In all cases, the MLP model reduces the error by nearly 50%. This demonstrates that the MLP model has better generalizability with limited labeled data, highlighting the necessity of using MLP.

4.10 Parameters and Latency

Table 3. System overhead comparison.

Model	Graph size	Model size	Train. time	Infer. time
WiMU	14.63 MB	0.95 MB	1,018 s	0.2 ms
GraFin	16.78 MB	0.18 MB	1,543 s	0.9 ms
WePos	-	3.1 MB	1.2×10^4 s	0.3 ms
MYRCJ	-	-	-	10.09 s

The parameters and latency of WiMU are demonstrated as follows. For 30 selected floors, the average size of the AP graphs is 14.63 MB. The total size of the model is 0.95MB, including GNN (0.68) MB and MLP (0.27 MB). We train our models on a single RTX 5880 Ada GPU. The average time required to construct the AP graphs is 922 seconds, whereas the average training time is 96 seconds. Compared with GraFin, the total size of the graph and the model of WiMU is smaller. It also requires the shortest training time among the three ML-based models and achieves the shortest inference time among all four models.

5 Case Study

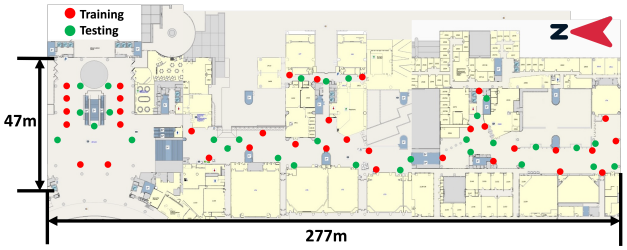
5.1 Implementation

We implement WiMU on a school campus to evaluate its performance in the real scenario. The building is highly complex due to rooms with varying layouts, spaces with irregular areas, and differences in elevation. As shown in Figure 11(b), we conduct the experiment in the major corridor that runs north and south through the building, and the shaded area is the region where we collect data and evaluate our system.

To implement a prototype of WiMU, we develop an Android application for data collection and real-time location inference. It can collect the step counter sensor, the rotation vector sensors (for deriving yaw), and the WiFi sensor data simultaneously with different frequencies. Due to the Android throttling strategy, the maximum WiFi sampling frequency is limited [62]. Therefore, we test seven different phone models from different brands including Samsung, Huawei, VIVO, Xiaomi, and OPPO. All the mobile phones except OPPO can set the WiFi sampling frequency up to $\frac{1}{15}$ Hz. Therefore, we set the sampling frequency of WiFi sensors to $\frac{1}{15}$ Hz. As for the

(a) Scenario photos.

(b) Unlabeled data collection region.



(c) Labeled data collection for training and testing.

Fig. 11. Scenario photos and floor plan of the campus.

IMU-based virtual sensors, including step counter and rotation vectors, we set the sampling frequency to 20 Hz to guarantee the accuracy of the PDR algorithm.

5.2 Case 1 - University Campus

We involve five volunteers to collect data for WiMU training and testing. Among the five volunteers, four are assigned to collect unlabeled RPs, and the last volunteer is asked to collect labeled RPs:

- **Unlabeled RP collections**. As shown in Figure 11(b), the four volunteers are asked to hold mobile devices and walk freely in the shaded area. Since the WiFi sampling frequency is $\frac{1}{15}$ Hz, it indicates that the app can collect an unlabeled RP with WiFi readings every 15 seconds. During the data collection process, volunteers are not required to do any annotations. It takes each of them around five minutes to complete one-time sampling.
- **Labeled RP collections**. To help the volunteer locate in reality easily, we select 65 notable landmarks such as doors, pillars, and parterres and annotate their coordinates on the floor map. A volunteer is asked to collect WiFi data at each location for one minute consecutively. Subsequently, labeled RPs are generated by aggregating the location annotations and WiFi data. As shown in Figure 11(c), we split the labeled RPs into a training dataset(35) and a testing dataset(30) to train and evaluate WiMU.

For the four volunteers responsible for the unlabeled RP collection, each of them spends 10 to 20 minutes completing 3 to 5 sampling tasks. For the volunteer responsible for the labeled RP collection, it takes around 1 hour to finish sampling since each labeled RP needs 1 minute of consecutive sampling. In total, we spend around 2 hours on data collection. As for the collected data, we obtain 2745 APs, 259 unlabeled RPs, and 65 labeled RPs(35 training + 30 testing).

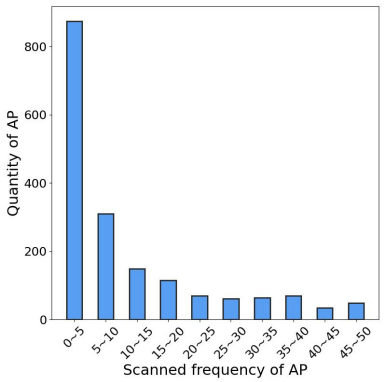


Fig. 12. Scanned frequencies of APs.

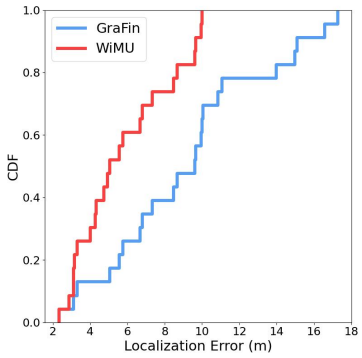


Fig. 13. CDF of localization errors on campus.

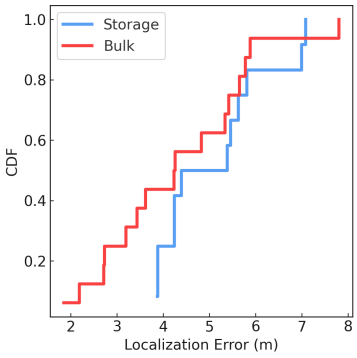


Fig. 14. CDF of localization errors in the warehouse.

After analyzing the SSIDs of collected APs, we find that some APs may actually be mobile hotspots. Since WiMU utilizes APs with static locations, we need to filter those hotspots due to the uncertainty of their locations. We observe that APs with fixed locations, such as campus networks, have a longer lifecycle compared to mobile hotspots and are more likely to be captured multiple times while sampling. Therefore, we count the scanned frequencies of each AP scanned and plot it as Figure 12, and keep the WiFi data only for APs that were scanned more than ten times. Among the collected 2745 APs, we keep 459 for the graph generation.

After the data cleaning stage, we build the AP-RP graph with the collected unlabeled RPs and refine it into an AP graph by measuring the average AP-to-AP distances. Subsequently, we train WiMU with the generated graph and augmented training dataset, then evaluate the performance on the testing dataset. The average localization error results in 5.807 meters. Besides, the best baseline(i.e., GraFin) achieves the average localization error in 9.217 meters. As shown in Figure 13, the CDF of localization errors illustrates that WiMU outperforms GraFin on localization accuracy and has more robust performance.

The effectiveness of the particle filter is evaluated and illustrated in Figure 15. In the figure, red circles represent the locations estimated by the particle filter, green circles indicate the ground-truth positions, and orange circles denote the predictions from the WiFi localization module. The numbers inside the circles denote the visiting order of each location. As shown in both trajectories, the incorporation of sensor fusion significantly improves localization accuracy by aligning the predicted path more closely with the ground truth.

5.3 Case 2 - Warehouse

WiMU is further evaluated in a real-world warehouse environment, as shown in Figure 16. The warehouse is divided into four functional zones: three storage areas and one bulk area. The storage zones are characterized by tall shelves approximately 15 meters high, whereas the bulk area features lower shelves ranging from 2 to 3 meters, reflecting two distinct spatial layouts. For evaluation, one representative area from each layout type is selected. Two volunteers collected data in the designated regions. In the storage area, 29 unlabeled trajectories and 59 labeled RPs were collected, while in the bulk area, 62 unlabeled trajectories and 82 labeled RPs were gathered. The entire data collection process took approximately two hours.

The labeled RPs were then split into training and testing sets, with 80% used for training and 20% for evaluation. Following the same methodology as in Case Study 1, we constructed the AP-RP graph and generated the corresponding representations. The model was fine-tuned using the training RPs, and the localization performance

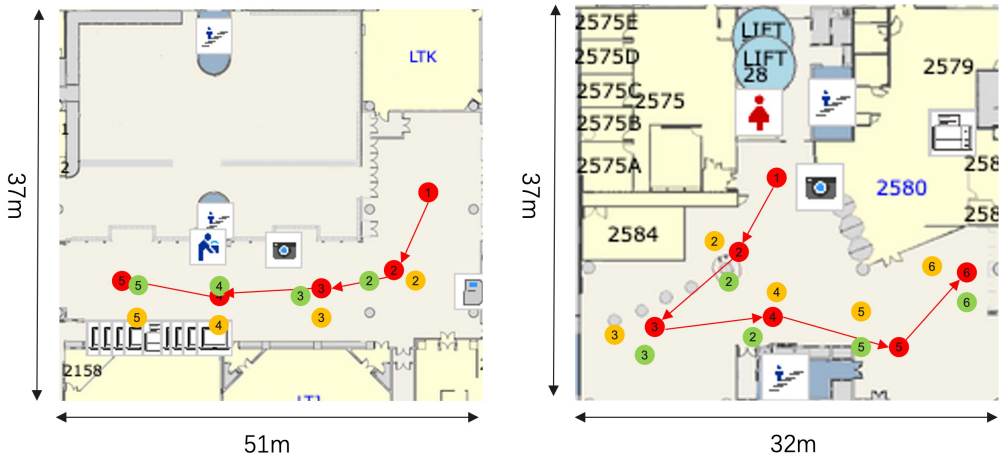


Fig. 15. Two real-world traces. Red circles represent the predictions of the particle filter, green circles indicate the ground-truth locations, and orange circles show the predictions from the WiFi localization module.

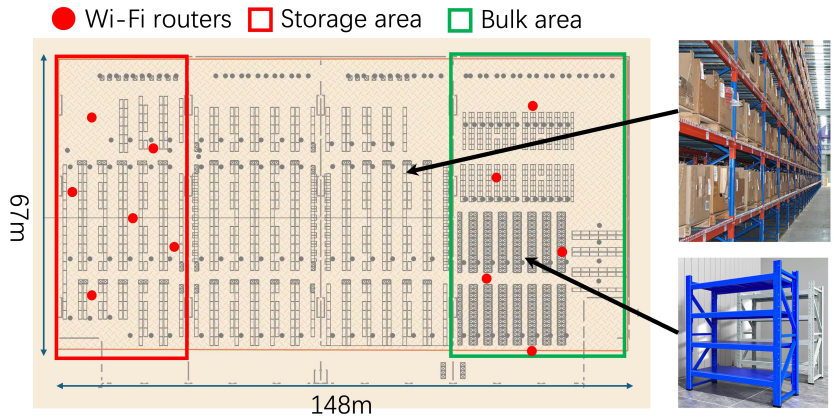


Fig. 16. The overview of the warehouse layout. The rough locations of routers are spotted. Different types of shelves are shown on the right

was assessed on the testing set. The cumulative distribution function (CDF) of localization errors is shown in Figure 14. WiMU achieves an average localization error of 4.61 m in the storage area and 5.22 m in the bulk area. The higher accuracy in the storage area is attributed to its more stable layout and wider WiFi coverage compared to the bulk area.

6 Conclusion

In this paper, we address a practical scenario for WiFi-based indoor localization, where the majority of fingerprint locations are unlabeled. To this end, we propose WiMU, a indoor localization framework that fuses WiFi and IMU data to learn effective spatial representations for accurate, real-time, and low-cost indoor localization. We

implement WiMU and evaluate it on the Microsoft Indoor Navigation Dataset, as well as through deployments in real-world environments. Extensive experimental results demonstrate that WiMU outperforms three state-of-the-art baselines and maintains robust performance across diverse deployment settings. By significantly reducing the cost and effort of system setup while preserving localization accuracy, WiMU facilitates scalable deployment of indoor localization systems.

Acknowledgments

We thank all reviewers for their insightful comments. This work is supported by the Global STEM Professorship Scheme of Hong Kong, the HKUST start up grant, and the Research Grants Council (RGC) General Research Fund (GRF) 16204224 and 16210425. Mo Li is the corresponding author.

References

[1] Jiazhi Ni, Fusang Zhang, Jie Xiong, Qiang Huang, Zhaoxin Chang, Junqi Ma, BinBin Xie, Pengsen Wang, Guangyu Bian, Xin Li, et al. Experience: Pushing indoor localization from laboratory to the wild. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 147–157, 2022.

[2] Bang Wang, Qiuyun Chen, Laurence T Yang, and Han-Chieh Chao. Indoor smartphone localization via fingerprint crowdsourcing: Challenges and approaches. *IEEE Wireless Communications*, 23(3):82–89, 2016.

[3] Wanlong Zhao, Shuai Han, Rose Qingyang Hu, Weixiao Meng, and Ziqing Jia. Crowdsourcing and multisource fusion-based fingerprint sensing in smartphone localization. *IEEE Sensors Journal*, 18(8):3236–3247, 2018.

[4] Dong Li, Baoxian Zhang, and Cheng Li. A feature-scaling-based k-nearest neighbor algorithm for indoor positioning systems. *IEEE Internet of Things Journal*, 3(4):590–597, 2015.

[5] Minh Tu Hoang, Yizhou Zhu, Brosnan Yuen, Tyler Reese, Xiaodai Dong, Tao Lu, Robert Westendorp, and Michael Xie. A soft range limited k-nearest neighbors algorithm for indoor localization enhancement. *IEEE Sensors Journal*, 18(24):10208–10216, 2018.

[6] Jin-Woo Jang and Song-Nam Hong. Indoor localization with wifi fingerprinting using convolutional neural network. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 753–758. IEEE, 2018.

[7] Mai Ibrahim, Marwan Torki, and Mustafa ElNainay. Cnn based indoor localization using rss time-series. In *2018 IEEE symposium on computers and communications (ISCC)*, pages 01044–01049. IEEE, 2018.

[8] Chaur-Heh Hsieh, Jen-Yang Chen, and Bo-Hong Nien. Deep learning-based indoor localization using received signal strength and channel state information. *IEEE access*, 7:33256–33267, 2019.

[9] Xudong Song, Xiaochen Fan, Chaocan Xiang, Qianwen Ye, Leyu Liu, Zumin Wang, Xiangjian He, Ning Yang, and Gengfa Fang. A novel convolutional neural network based indoor localization framework with wifi fingerprinting. *IEEE access*, 7:110698–110709, 2019.

[10] Yuming Hu, Feng Qian, Zhimeng Yin, Zhenhua Li, Zhe Ji, Yeqiang Han, Qiang Xu, and Wei Jiang. Experience: Practical indoor localization for malls. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 82–93, 2022.

[11] Zayan El Khaled, Wessam Ajib, and Hamid Mcheick. Log distance path loss model: Application and improvement for sub 5 ghz rural fixed wireless networks. *IEEE Access*, 10:52020–52029, 2022.

[12] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[13] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F - Radar and Signal Processing*, 140(2):107–113, 1993.

[14] Bin Yu AlphaJi, inversion EvanSong1220, Qiang Xu Kumius, and Yuanchao Shu. Indoor location and navigation, 2021.

[15] Huthaifa Obeidat, Wafa Shuaieb, Omar Obeidat, and Raed Abd-Alhameed. A review of indoor localization techniques and wireless technologies. *Wireless Personal Communications*, 119:289–327, 2021.

[16] Alwin Poulose and Dong Seog Han. Uwb indoor localization using deep learning lstm networks. *Applied Sciences*, 10(18):6290, 2020.

[17] Jenny Röbesaat, Peilin Zhang, Mohamed Abdelaal, and Oliver Theel. An improved ble indoor localization with kalman-based fusion: An experimental study. *Sensors*, 17(5):951, 2017.

[18] Xiansheng Guo, Sihua Shao, Nirwan Ansari, and Abdallah Khreishah. Indoor localization using visible light via fusion of multiple classifiers. *IEEE photonics journal*, 9(6):1–16, 2017.

[19] Cem Sertatıl, Mustafa A Altınkaya, and Kosai Raoof. A novel acoustic indoor localization system employing cdma. *Digital Signal Processing*, 22(3):506–517, 2012.

[20] Wonho Kang and Youngnam Han. Smartpdr: Smartphone-based pedestrian dead reckoning for indoor localization. *IEEE Sensors journal*, 15(5):2906–2916, 2014.

[21] Sinno Jialin Pan, Vincent Wenchen Zheng, Qiang Yang, and Derek Hao Hu. Transfer learning for wifi-based indoor localization. In *Association for the advancement of artificial intelligence (AAAI) workshop*, volume 6. The Association for the Advancement of Artificial

Intelligence Palo Alto, 2008.
[22] Chouchang Yang and Huai-Rong Shao. Wifi-based indoor positioning. *IEEE Communications Magazine*, 53(3):150–157, 2015.
[23] Guenther Retscher. Fundamental concepts and evolution of wi-fi user localization: An overview based on different case studies. *Sensors*, 20(18):5121, 2020.
[24] Veli Ilci, R Metin Alkan, V Engin Gülal, and Huseyin Cizmeci. Trilateration technique for wifi-based indoor localization. *ICWMC 2015*, page 36, 2015.
[25] Xianan Zhang, Yu Zhang, Guanghua Liu, and Tao Jiang. Autoloc: Toward ubiquitous aoa-based indoor localization using commodity wifi. *IEEE Transactions on Vehicular Technology*, 72(6):8049–8060, 2023.
[26] Leor Banin, U Schtzberg, and Yuval Amizur. Next generation indoor positioning system based on wifi time of flight. In *Proceedings of the 26th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2013)*, pages 975–982, 2013.
[27] Navneet Singh, Sangho Choe, and Rajiv Punmiya. Machine learning based indoor localization using wi-fi rssi fingerprints: An overview. *IEEE Access*, 9:127150–127174, 2021.
[28] Minhui Luo, Jin Zheng, Wei Sun, and Xing Zhang. Wifi-based indoor localization using clustering and fusion fingerprint. In *2021 40th Chinese Control Conference (CCC)*, pages 3480–3485. IEEE, 2021.
[29] Yanzhao Wang, Chundi Xiu, Xuanli Zhang, and Dongkai Yang. Wifi indoor localization with csi fingerprinting-based random forest. *Sensors*, 18(9):2869, 2018.
[30] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.
[31] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N Padmanabhan. Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 173–184, 2010.
[32] Anshul Rai, Krishna Kant Chintalapudi, Venkata N Padmanabhan, and Rijurekha Sen. Zee: Zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 293–304, 2012.
[33] Michael Hardegger, Daniel Roggen, and Gerhard Tröster. 3d actionslam: Wearable person tracking in multi-floor environments. *Personal and ubiquitous computing*, 19(1):123–141, 2015.
[34] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, and Xiaojie Guo. Graph neural networks: foundation, frontiers and applications. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4840–4841, 2022.
[35] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. Meta-gnn: On few-shot node classification in graph meta-learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2357–2360, 2019.
[36] Chen Cai and Yusu Wang. A simple yet effective baseline for non-attributed graph classification. *arXiv preprint arXiv:1811.03508*, 2018.
[37] Jonathan Godwin, Michael Schaarschmidt, Alexander Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Veličković, James Kirkpatrick, and Peter Battaglia. Simple gnn regularisation for 3d molecular property prediction & beyond. *arXiv preprint arXiv:2106.07971*, 2021.
[38] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, et al. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93, 2022.
[39] Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. Graph neural networks and their current applications in bioinformatics. *Frontiers in genetics*, 12:690049, 2021.
[40] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328, 2023.
[41] Tianrui Chen, Xinruo Zhang, Minglei You, Gan Zheng, and Sangarapillai Lambotharan. A gnn-based supervised learning framework for resource allocation in wireless iot networks. *IEEE Internet of Things Journal*, 9(3):1712–1724, 2021.
[42] Shengjie Liu, Jia Guo, and Chenyang Yang. Multidimensional graph neural networks for wireless communications. *IEEE Transactions on Wireless Communications*, 2023.
[43] Xiao Xu, You Lu, and Qiming Fu. Applying graph neural network in deep reinforcement learning to optimize wireless network routing. In *2021 Ninth International Conference on Advanced Cloud and Big Data (CBD)*, pages 218–223. IEEE, 2022.
[44] Yaya Etiabi, Eslam Eldeeb, Mohammad Shehab, Wafa Njima, Hirley Alves, Mohamed-Slim Alouini, and El Mehdi Amhoud. Metagraphloc: A graph-based meta-learning scheme for indoor localization via sensor fusion, 2024.
[45] Yuanyi Chen, Minyi Guo, Jiaxing Shen, and Jiannong Cao. Graphloc: a graph-based method for indoor subarea localization with zero-configuration. *Personal Ubiquitous Comput.*, 21(3):489–505, June 2017.
[46] Han Zheng, Yan Zhang, Lan Zhang, Hao Xia, Shaojie Bai, Guobin Shen, Tian He, and Xiangyang Li. Grafin: An applicable graph-based fingerprinting approach for robust indoor localization. In *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 747–754. IEEE, 2021.
[47] Theodore S Rappaport. *Wireless communications: principles and practice*. Cambridge University Press, 2024.
[48] Haroon Rashid and Ashok Kumar Turuk. Dead reckoning localization technique for mobile wireless sensor networks. *IET Wireless Sensor Systems*, 5(2):87–96, 2015.

[49] Indoor Location Competition. Indoor location and navigation competition 2020. https://github.com/location-competition/indoor-location-competition-20.
[50] Google. Position sensors | android developers, 2023. Accessed: 2023-10-15.
[51] Bent Fuglede and Flemming Topsoe. Jensen-shannon divergence and hilbert space embedding. In *International symposium onInformation theory, 2004. ISIT 2004. Proceedings.*, page 31. IEEE, 2004.
[52] Frank Nielsen. On a generalization of the jensen–shannon divergence and the jensen–shannon centroid. *Entropy*, 22(2):221, 2020.
[53] Patrick Billingsley. *Probability and measure*. John Wiley & Sons, 2017.
[54] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
[55] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
[56] Randal Douc and Olivier Cappé. Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69. Ieee, 2005.
[57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
[58] Baoshen Guo, Weijian Zuo, Shuai Wang, Wenjun Lyu, Zhiqing Hong, Yi Ding, Tian He, and Desheng Zhang. Wepos: Weak-supervised indoor positioning with unlabeled wifi for on-demand delivery. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(2):1–25, 2022.
[59] Yuming Hu, Xiubin Fan, Zhimeng Yin, Feng Qian, Zhe Ji, Yuanchao Shu, Yeqiang Han, Qiang Xu, Jie Liu, and Paramvir Bahl. The wisdom of 1,170 teams: Lessons and experiences from a large indoor localization competition. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '23, New York, NY, USA, 2023. Association for Computing Machinery.
[60] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
[61] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.
[62] Google. Android wi-fi scanning overview. https://developer.android.com/develop/connectivity/wifi/wifi-scan. Accessed: 2024-12-07.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009