Received July 31, 2019, accepted August 17, 2019, date of publication September 16, 2019, date of current version September 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2941657

Unsupervised Learning Techniques for Trilateration: From Theory to Android APP Implementation

JEONGSIK CHOI[1], YANG-SEOK CHOI[2], AND SHILPA TALWAR[1]
[1]Intel Labs, Intel Corporation, Santa Clara, CA 95054, USA
[2]Intel Labs, Intel Corporation, Hillsboro, OR 97124, USA

Corresponding author: Jeongsik Choi (jeongsik.choi@intel.com)

ABSTRACT Because the characteristics of wireless propagation channels (especially indoor channels) are too diverse and complex, the distance estimation strategy of range-based positioning techniques should adaptively change depending on the environment. In this paper, we study unsupervised learning techniques that efficiently do this without human intervention. As users simply move around an area of interest with mobile devices, the proposed method autonomously learns the characteristics of the surrounding environments and changes the ranging strategy accordingly. To this end, we use either model-based or neural network (NN)-based ranging modules for estimating the distance from neighboring anchor nodes, calculate the position of the devices using trilateration techniques, and define cost functions that indirectly evaluate the accuracy of the ranging module based on the trilateration results. Moreover, by assigning a unique trainable variable to each device, the proposed method is also able to compensate for different characteristics between devices without ground truth data. The performance of the proposed method is verified with a real-time location tracking application using received signal strength (RSS) measurements from conventional Wi-Fi access points (APs) or round trip time (RTT) measurements from APs that support the fine timing measurement (FTM) protocol. In cases where a model-based ranging module is used, the proposed method closely achieves the benchmark performance, which perfectly optimizes all the trainable variables on the test data. If NNs are adopted in the ranging module, the proposed method even outperforms the benchmark and achieves an average positioning accuracy of up to 2.397 m using RSS measurements, and up to 1.547 m using RTT measurements under the 40 MHz bandwidth configuration.

INDEX TERMS Fine timing measurement (FTM), neural network, positioning, trilateration, unsupervised learning.

I. INTRODUCTION

Precise location information is essential for several applications such as navigation systems, autonomous driving, vehicle/asset tracking, and other location-based services. For this reason, positioning techniques have been constantly attracting attention over the last few decades. The global navigation satellite system (GNSS) is one of the most promising positioning solutions in outdoor environments, where direct paths from satellites can be secured. In indoor environments, however, there is no single solution, and a variety of approaches have been introduced in the literatures. One approach is to exploit the wireless communication technologies such as cellular [1]–[4], Wi-Fi [5]–[16], Bluetooth [17]–[19], ultra wide band (UWB) [20]–[22], and near field communication (NFC) [23], [24]. Moreover, sensors such as inertial measurement units (IMUs) and magnetometers [25]–[29], vision sensors [30], [31], and foot-mounted sensors [32]–[35] are also widely used for positioning purposes.

In this study, we focus on the Wi-Fi-based approach, because many indoor environments already have a sufficient number of Wi-Fi access points (APs) installed. Therefore, if a mobile device can measure distances from at least three APs, its location can be easily obtained using trilateration techniques [7], [10], [11], [13], [14], [18]. One of the simplest ways to measure distance is to use received signal strength (RSS), which generally decays with distance from the transmitter [36], [37]. However, each environment has

The associate editor coordinating the review of this manuscript and approving it for publication was Choon Ki Ahn.

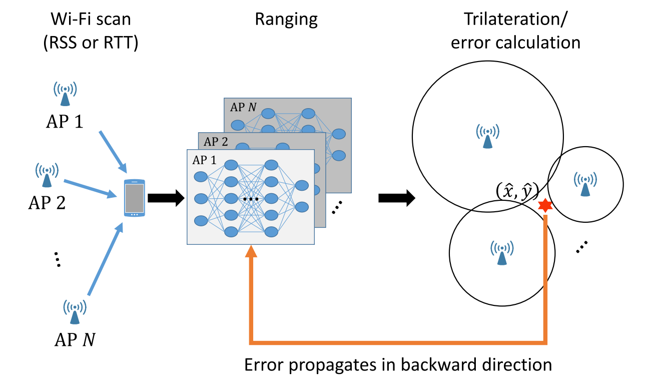


FIGURE 1. Overview of the proposed unsupervised learning framework. Based on the outputs of the ranging module (either model or NN-based), the coordinates of a device can be obtained. With positioning results, cost functions are defined and then used for training.

unique propagation characteristics that should be investigated for accurate ranging results. In addition, RSS is also affected by many factors such as the presence of the line-of-sight (LOS) path, the structure and materials of the surrounding environments, and shadowing from obstacles/human bodies [38]–[41]. For this reason, accurately measuring distance with the RSS is challenging and results in the poor positioning quality.

To overcome the limitation of the distance measurement using RSS, Wi-Fi fingerprinting techniques have been extensively studied in the literatures [5], [6], [9], [12], [17]. This method does not rely on the distance measurement; rather, it tabulates measured RSS from neighboring APs at the selected location as the signatures for finding the location. Once a database of RSS measurements (also known as a radio map) is prepared, the location of a device can be found by comparing current RSS measurements with the radio map. However, it takes time and effort to build a unique radio map for each environment. For instance, a separate radio map is required for each floor, even in the same building.

To support Wi-Fi-based positioning, the IEEE 802.11-2016 standard (also known as 802.11REVmc) introduced an improved ranging protocol called fine timing measurement (FTM) that measures the distance between two Wi-Fi devices based on the round trip time (RTT) of wireless packets [42]. Although the FTM protocol requires a calibration process to correct distorted outputs [15], [43], once calibration is done, it produces relatively accurate ranging results compared to RSS. With well-calibrated distance measurements, the location of a mobile device can be simply obtained by collecting multiple distance measurements from nearby anchor APs whose coordinates are known [13]–[16].

The main objective of this study is to minimize human intervention in the deployment of positioning solutions. From this perspective, we believe that a range-based positioning approach that automatically optimizes the ranging strategy is preferable to the fingerprinting methods. To this end, we propose an unsupervised learning framework, as illustrated in Fig. 1, where a model-based or a neural network (NN)-based ranging module adjusts the trainable parameters in the module depending on the environment. Irrespective of how well the parameters are optimized, the ranging module always produces distance estimates, and consequently, the coordinates of a device can be obtained. Based on the estimated coordinates, the cost functions that indirectly evaluate the accuracy of the ranging module are defined and then utilized for training.

The contributions of this paper are summarized as follows:

1) We design cost functions that do not necessarily require labeled data. For this reason, training data can be easily obtained even when users are using location services, which greatly reduces human intervention. With cost functions, the proposed method autonomously learns how to optimize trainable parameters. For example, a calibration process for the FTM protocol can be done automatically, instead of manual calibration in [43].
2) To efficiently deal with various types of mobile devices, the proposed method shares the same ranging module for all devices and compensates each device's unique characteristics with trainable variables. This allows us to obtain a single precise ranging module using rich training data collected from multiple devices. As an interesting result, collecting only a few ground truth coordinates using a device actually improves the positioning accuracy of all devices.
3) We implement a real-time Android application to collect training data and to verify the performance of the proposed method. We also test various mobile devices from different manufactures. With existing IEEE 802.11ac APs, we verify the performance of the RSS-based positioning method. In addition, we install 802.11-2016 capable APs to evaluate the performance of the RTT-based positioning method.

The rest of the paper is organized as follows. In the next section, we briefly introduce related works. In Section III, we discuss the ranging module, offset compensation, and positioning methods. In Section IV, we introduce cost functions for training. The experiment results are presented in Section V, followed by the conclusion.

Notation: Boldface letters are used for matrices and vectors. $\mathbf{A} \in \mathbb{R}^{N\times M}$ represents an $N \times M$ real matrix (or a vector). $(\cdot)^{-1}$ and $(\cdot)^T$ are the inverse and transpose operators, respectively. $\mathbf{A} = \text{diag}(a_1, \ldots, a_N) \in \mathbb{R}^{N\times N}$ is the diagonal matrix whose diagonal elements are $a_1, \ldots, a_N$. $\mathbf{I}_N \in \mathbb{R}^{N\times N}$ is the identity matrix, $\mathbf{0}_N \in \mathbb{R}^{N\times 1}$ is the zero vector, $||\mathbf{a}|| = \sqrt{\mathbf{a}^T\mathbf{a}}$ is the $l2$-norm of a vector $\mathbf{a} \in \mathbb{R}^{N\times 1}$, and $E[\cdot]$ is the expectation operator.

II. RELATED WORKS

Distance estimation in Wi-Fi systems is generally performed with RSS and RTT measurements. As we mentioned in the introduction, the RTT-based ranging protocol FTM was recently added to the IEEE 802.11 standard, and thus, not all devices and APs support this feature. The ranging and positioning performance achieved using the FTM protocol has been verified in [13]–[15]. In addition, a supervised learning approach for training NNs to estimate distances from raw channel state information (CSI) measurements of FTM packets is introduced in [16].

On the other hand, every device can measure the RSS from each AP by monitoring the beacon transmission and using it to estimate distance. The pathloss model describes the relationship between the RSS and distance using two parameters: the pathloss exponent and signal strength at a reference distance [36], [37]. Alternatively, the distance can be calculated as a polynomial of RSS [10]. This method potentially learns more flexible curves between the RSS and distance if a high order polynomial is used. To optimize parameters in the pathloss model depending on the environment, joint optimization of parameters and the coordinates of the device has been studied in [44]–[46].

As RSS is highly dominated by the existence of a direct path between two nodes, it is also important to identify channel conditions, such as LOS or non-LOS (NLOS) conditions, to achieve better ranging accuracy by applying different strategies accordingly. The propagation condition can be inferred from the time-varying pattern of the RSS [47] or the difference in RSS measurements between multiple frequency bands (e.g., 2.4 and 5 GHz) [48]. In cases where CSI is available, many handcrafted features can be extracted to identify channel condition [49]–[51]. In addition, NNs are also widely used to extract useful features to identify channel conditions [52], [53].

III. SYSTEM MODELS

We use Wi-Fi AP as the anchor node for positioning. Therefore, we use the terms AP and anchor node interchangeably. First, we focus on a general ranging procedure between an AP and a device. The ranging module produces a distance estimate and its standard deviation from the input data. Therefore, the relationship between the input and output of the ranging module $\mathbf{r}(\cdot)$ is expressed as a parameterized function as follows:

$$[\hat{d}, \hat{s}]^T = \mathbf{r}(\mathbf{x}; \Theta), \tag{1}$$

where $\mathbf{x}$ is an input vector, Θ is the set of all trainable parameters in the module, $\hat{d}$ is the distance estimate, and $\hat{s}$ is the expected standard deviation of $\hat{d}$.

The input vector can contain any information obtained from the AP. However, we use Android devices for the experiments and limited information are available. For instance, the RSS value is the only piece of information that we can obtain from conventional APs. In this case, we simply use $\mathbf{x} = RSS$ as the input of the ranging module.[1]

If an AP and a device support the FTM protocol, we can obtain RTT-related measurements. Generally, the FTM protocol performs multiple ranging procedures for a single ranging request and reports the average and standard deviation of multiple ranging results. In addition, it also returns the RSS of FTM packets. Therefore, the input vector can be expressed by $\mathbf{x} = [d^{FTM}, s^{FTM}, RSS^{FTM}]^T$, where each element represents the distance, standard deviation, and RSS reported from the FTM protocol, respectively.

The ranging module can be implemented using a model or an NN. Each implementation will be discussed in the next subsections. In addition, we also introduce a simple way to compensate for different behaviors between various APs and devices. At the end of this section, we describe trilateration techniques that calculate the coordinates of a device by using the ranging results from multiple APs in the vicinity.

A. RANGING MODELS

To estimate distance using the RSS, the pathloss model has been widely used [7], [8], [18]. According to the model, RSS is decided by

$$RSS = RSS(d_0) - 10\eta \log_{10}\left(\frac{d}{d_0}\right) + X, \tag{2}$$

where $RSS(d_0)$ is the signal strength measured at a reference distance d_0 (usually 1 m), η is the pathloss exponent, and X represents additional losses including shadowing. Using this model, the distance estimate is derived as

$$\hat{d}^{RSS,PL} = d_0 10^{\frac{RSS(d_0)-RSS}{10\eta}}. \tag{3}$$

This model consists of two trainable parameters.

Alternatively, the distance can be expressed as a polynomial of RSS [10]. In this work, we use a quadratic polynomial expressed by

$$\hat{d}^{RSS,poly} = c_2 RSS^2 + c_1 RSS + c_0, \tag{4}$$

where c_2, c_1 and c_0 are coefficients of the polynomial that need to be chosen appropriately depending on the environment. Regardless of which model is used, we can expect that the ranging error is proportional to the distance estimate. Therefore, the expected standard deviation of the distance estimate is simply modeled as

$$\hat{s} = \beta \hat{d}, \tag{5}$$

where β is a proportional constant, and $\hat{d}$ is the distance estimate obtained using equation (3) or (4).

In cases in which the FTM protocol is supported, we can use the distance and standard deviation reported by the protocol. However, according to the results in [15] and our results as well, distance measurements acquired using the FTM protocol have a remarkable bias that needs to be calibrated. Therefore, we use a simple offset model to compensate for the bias:

$$\hat{d}^{RTT} = \max(d^{FTM} + \phi, 0), \tag{6}$$

where ϕ is the distance measurement offset (due to the AP or the device or both). The $\max(\cdot)$ function is used to ensure that the distance estimate is always non-negative. For the

[1]This work uses RSS obtained from a single frequency band. If RSS from multiple frequency bands or multiple antennas, or even CSI are available, we can include all information in the input vector.

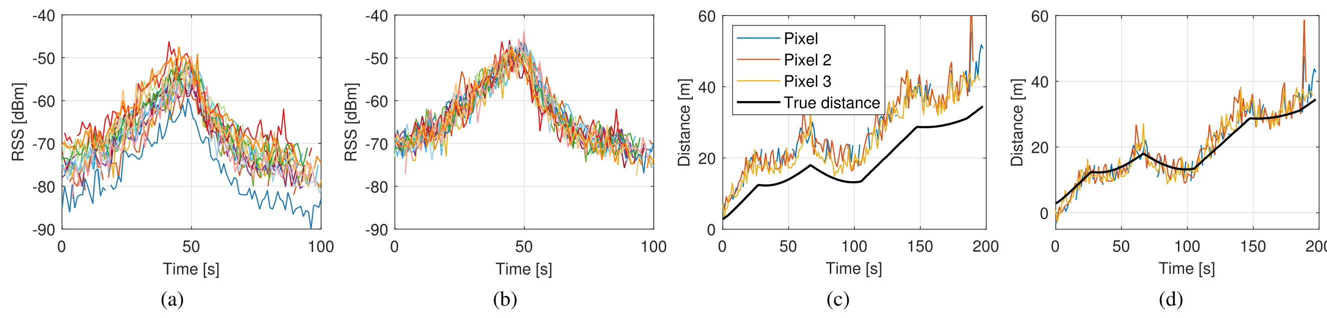


FIGURE 2. RSS/RTT measurements of various mobile devices moving in the same path: (a) Raw RSS measurements of 15 devices, (b) RSS measurements after offset compensation, (c) raw distance measurements of Google Pixel series using the FTM protocol, and (d) calibrated distance measurements with offsets.

TABLE 1. Summary of trainable parameters in each model.

Source	Distance	Standard deviation	Parameters
RSS	Eq. (3)	Eq. (5)	$\Theta = \{RSS(d_0), \eta, \beta\}$
	Eq. (4)	Eq. (5)	$\Theta = \{c_2, c_1, c_0, \beta\}$
RTT	Eq. (6)	s^{FTM} or Eq. (5)	$\Theta = \{\phi\}$ or $\{\phi, \beta\}$

standard deviation of the distance estimate, we can use either the reported value as $\hat{s} = s^{FTM}$ or the proportional model in equation (5). The model-based ranging modules have relatively few parameters as summarized in Table 1.

B. RANGING USING NEURAL NETWORKS

Instead of using models, we can apply NNs for ranging. In general, NNs are able to learn more flexible relationships between inputs and outputs with a large number of trainable parameters. In this work, we mainly exploit simple fully-connected (FC) layers. We assume that there are L hidden layers between the input and output layers, and the l-th layer has D_l hidden nodes. Then, the activation vector of the l-th layer is computed as

$$\mathbf{a}_l = \sigma_l(\mathbf{W}_l \mathbf{a}_{l-1} + \mathbf{b}_l), \quad l = 1, \ldots, L, \tag{7}$$

where $\mathbf{a}_l \in \mathbb{R}^{D_l \times 1}$ is the activation vector, $\mathbf{W}_l \in \mathbb{R}^{D_l \times D_{l-1}}$ is a weight matrix, $\mathbf{b}_l \in \mathbb{R}^{D_l \times 1}$ is a bias vectors, and $\sigma_l(\cdot)$ is a non-linear activation function. To ensure that equation (7) covers the input layer, we simply put $\mathbf{a}_0 = \mathbf{x}$.

The last hidden layer produces two outputs, which are the distance estimate and its standard deviation. They are represented by

$$\begin{aligned} \hat{d} &= d_{max}\sigma(\mathbf{W}^{dist}\mathbf{a}_L + b^{dist}), \\ \hat{s} &= s_{max}\sigma(\mathbf{W}^{std}\mathbf{a}_L + b^{std}), \end{aligned} \tag{8}$$

where $\sigma(\cdot)$ represents the sigmoid activation function, $\mathbf{W}^{dist}$, $\mathbf{W}^{std} \in \mathbb{R}^{1 \times D_L}$ are weight matrices, and b^{dist}, b^{std} are scalar biases. The constants d_{max} and s_{max} are used to set the upper bound of each output. Alternatively, a rectified linear unit (ReLU) can be applied for output activation.

In addition to the FC layers, we can apply various structures of NNs. In particular, we use a recurrent NN (RNN) with long short-term memory (LSTM) similar to the one used in [52]. Regardless of which NN structure is applied to extract features from the input layer, the outputs of the ranging module can be calculated from the last hidden layer as shown in equation (8).

TABLE 2. List of Android devices used in the experiments.

Device #	Model name	Release date	Version
1	LG Nexus 5	2013/10	6.0.1
2	Sony Xperia Z2	2014/03	6.0.1
3	LG G3	2014/05	6.0
4	Samsung Galaxy Note 4	2014/10	6.0.1
5	HTC One M9	2015/04	7.0
6	Samsung Galaxy Note 5	2015/08	7.0
7	Huawei Nexus 6P	2015/09	7.1.1
8	Samsung Galaxy S7	2016/03	8.0
9	HTC 10	2016/04	8.0
10	LG G6	2017/03	7.0
11	Samsung Galaxy Note 8	2017/08	7.1.1
12	Motorola Moto X4	2017/09	7.1.1
13	OnePlus 5T	2017/11	7.1.1
14	LG V30+	2018/03	8.0.0
15	LG G7	2018/05	8.0.0
16	Google Pixel [a]	2016/10	9
17	Google Pixel 2 [a]	2017/10	9
18	Google Pixel 3 [a]	2018/10	9

[a]Google Pixel series supports FTM functionality on Android 9.

C. OFFSET COMPENSATION

There are a variety of mobile devices in the world, and each has unique characteristics. Fig. 2(a) shows the raw RSS measurements of 15 different devices moving in the same path (the device list is summarized in the Table 2). The variation patterns are almost similar except for the offsets. If we add an appropriate offset to each device, the RSS curves overlap as shown in Fig. 2(b). A similar phenomenon is observed in Fig. 2(c) and 2(d). The raw distance measurements of Google Pixel series using the FTM protocol have relative offsets between devices as well as absolute offsets from the true distances.

The different characteristics of multiple devices can be addressed by deploying a separate ranging module for each device. In this case, each device should collect a sufficient amount of training data to obtain accurate parameters. Instead, if all devices share the same ranging module and each device's characteristics are compensated for appropriately, rich training data collected from multiple devices can be used to obtain a single precise ranging module.

For this reason, we introduce additional trainable variables for each AP and device. We denote $\mathbf{x}_{n,k}$ as the input data for the ranging module to estimate the distance and its standard deviation between the n-th AP and the k-th device. Instead of directly feeding this raw measurement into the shared ranging module, we can feed it into a modified ranging module $\tilde{\mathbf{r}}(\cdot)$ defined by

$$[\hat{d}_{n,k}, \hat{s}_{n,k}]^T = \tilde{\mathbf{r}}(\mathbf{x}_{n,k}; \tilde{\Theta}) = \mathbf{r}(\mathbf{x}_{n,k} + \mathbf{x}_n^{AP} + \mathbf{x}_k^{dev}; \Theta), \quad (9)$$

where $\mathbf{x}_n^{AP}$ and $\mathbf{x}_k^{dev}$ represent the offsets of the n-th AP and the k-th device, respectively. The offsets have the same dimension with the input layer. The modified ranging module has a larger set of trainable parameters, i.e., $\tilde{\Theta} = \Theta \cup \{\mathbf{x}_n^{AP} \text{ for } \forall n\} \cup \{\mathbf{x}_k^{dev} \text{ for } \forall k\}$.

D. TRILATERATION WITH OUTPUTS OF THE RANGING MODULE

In this subsection, we focus on trilateration-based positioning techniques for a mobile device. For this reason, we omit the index of the device. We consider a two-dimensional space, and denote $\mathbf{z} = [x, y]^T$ as the coordinates of the device.

The coordinates of the device can be estimated using the distance and standard estimates from N nearby anchor nodes whose coordinates are known. Therefore, the estimated coordinates are expressed as a function as follows:

$$\hat{\mathbf{z}} = \mathbf{p}(\mathbf{m}_1, \ldots, \mathbf{m}_N), \quad (10)$$

where $\mathbf{m}_n = [x_n, y_n, \hat{d}_n, \hat{s}_n]^T$ is a vector consisting of the x and y coordinates of the n-th anchor node, distance and standard deviation estimates from this node.

As we mentioned in Fig. 1, the proposed method defines cost functions using the estimated coordinates. Therefore, to calculate the gradient of cost functions, it is necessary to know the impact of slight variations in each trainable parameter on the estimated coordinates first. Let $\theta \in \tilde{\Theta}$ denote a parameter. Then, we have the following relationship using the chain rule:

$$\frac{\partial \hat{\mathbf{z}}}{\partial \theta} = \sum_{n=1}^{N} \frac{\partial \mathbf{p}(\mathbf{m}_1, \ldots, \mathbf{m}_N)}{\partial \mathbf{m}_n} \frac{\partial \mathbf{m}_n}{\partial \theta}. \quad (11)$$

One necessary condition for obtaining the above derivative is that $\mathbf{p}(\cdot)$ is differentiable with respect to every argument. For this reason, we consider linear trilateration methods that calculate the estimated coordinates using matrix multiplications only [54]–[57].

As an example, the linear least-square (LS) method [56] estimates the coordinates of the device as

$$\hat{\mathbf{z}}^{LS} = \mathbf{p}^{LS}(\mathbf{m}_1, \ldots, \mathbf{m}_N) = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{q}, \quad (12)$$

where $\mathbf{A} \in \mathbb{R}^{(N-1)\times 2}$ is the matrix and $\mathbf{q} \in \mathbb{R}^{(N-1)\times 1}$ is the vector. Both are defined by

$$\mathbf{A} = \begin{bmatrix} 2x_2 - 2x_1 & 2y_2 - 2y_1 \\ \vdots & \vdots \\ 2x_N - 2x_1 & 2y_N - 2y_1 \end{bmatrix},$$
$$\mathbf{q} = \begin{bmatrix} x_2^2 - x_1^2 + y_2^2 - y_1^2 - \hat{d}_2^2 + \hat{d}_1^2 \\ \vdots \\ x_N^2 - x_1^2 + y_N^2 - y_1^2 - \hat{d}_N^2 + \hat{d}_1^2 \end{bmatrix}. \quad (13)$$

The LS method does not take into account the standard deviation of the distance estimate.

To improve the positioning accuracy by considering the different error behaviors of each distance estimate, the weighted linear least-square (WLS) method adds a weight matrix to the formulation of the LS method. According to [56], the result using the WLS method is given by

$$\hat{\mathbf{z}}^{WLS} = \mathbf{p}^{WLS}(\mathbf{m}_1, \ldots, \mathbf{m}_N) = (\mathbf{A}^T\mathbf{B}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B}\mathbf{q}, \quad (14)$$

where $\mathbf{A}$ and $\mathbf{q}$ are as previously defined, and $\mathbf{B} \in \mathbb{R}^{(N-1)\times(N-1)}$ represents the matrix whose j-th diagonal element is $var(\hat{d}_{j+1}^2)+var(\hat{d}_1^2)$ for $j = 1, \ldots, N-1$, and all other elements are $var(\hat{d}_1^2)$. Here, $var(\cdot)$ represents the variance, and $var(\hat{d}^2)$ is usually approximated as $var(\hat{d}^2) = \hat{d}\hat{s}^2$ [57].

Similar to these linear methods, the proposed method is also compatible with the Kalman filter, which estimates the hidden state (i.e., the coordinates of the device in this study) using matrix operations. In particular, we exploit the extended Kalman filter (EKF) which is able to deal with non-linear measurement model. The details of the EKF procedure are summarized in the Appendix. One difference between the EKF procedure and the linear methods is that it uses previous results to estimate the current state. For this reason, we slightly modify equation (10) as

$$\hat{\mathbf{z}}^{EKF}(i) = \mathbf{p}^{EKF}(\hat{\mathbf{z}}^{EKF}(i-1), \mathbf{m}_1(i), \ldots, \mathbf{m}_N(i)), \quad (15)$$

where $\hat{\mathbf{z}}^{EKF}(i)$ represents the estimated coordinates at time step i. And $\mathbf{m}_n(i)$ is as defined in equation (10), where the elements are obtained with respect to the n-th anchor node at time step i. Because the output of $\mathbf{p}^{EKF}(\cdot)$ is linearly related to all arguments, we can derive its derivative similar to equation (11).

IV. LEARNING TECHNIQUES

A. OVERVIEW

Supervised learning is the most reliable approach for optimizing the parameters in the ranging module, but requires collection of the true coordinates of mobile devices when measurements are taken. To this end, system operators may collect true coordinates at selected positions with their mobile

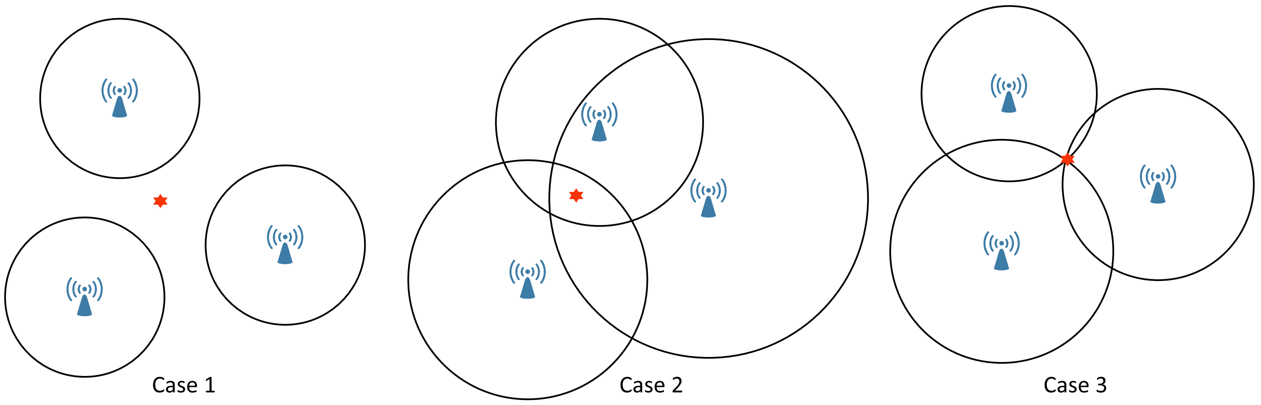


FIGURE 3. Intuition for unsupervised learning. If circles do not intersect at one point, it means that there are errors in the distance estimates.

devices. However, it takes too much time to collect a sufficient amount of training data, and there is no guarantee that the acquired training data are valid for devices other than those used by service operators.

However, the collection of ground truth data is not required for unsupervised learning, and thus, the training data can be obtained even while users are using location services. In this way, rich training data can be easily collected from various mobile devices. One of the biggest challenges of unsupervised learning is that it is difficult to know whether the ranging module is working properly or not. Therefore, we need to find a way to infer the accuracy of the module.

Fig. 3 illustrates the main intuition for unsupervised learning. The radius of each circle represents the distance estimate from each anchor node. In cases 1 and 2, the circles do not intersect at a single point. This means that there are errors in distance estimates. On the other hand, the circles intersect at exactly one point in the last case. Even though this intersection point may not be the true coordinates of the device, if case 3 is satisfied for every training data, we can assume that the ranging module is accurate.

B. COST FUNCTIONS FOR UNSUPERVISED LEARNING

Even if the ranging module is not well-trained, it always produces distance and standard deviation estimates for nearby APs. Thus, we can compute the estimated trajectory of a device using any trilateration techniques discussed in the previous section. The main role of cost functions is to evaluate the validity of an estimated trajectory so that the ranging module adjusts its parameters to produce a more plausible trajectory.

In this section, we focus on the training data obtained from a single device. If multiple devices are involved in collecting training data, the total cost for the training is simply defined as the sum of each individual device's cost. We assume that the training data are collected for T consecutive time steps, and let $\hat{\mathbf{z}}(i)$ and $\mathbf{z}_n(i)$ represent the estimated coordinates of the device and the coordinates of the n-th anchor node at time step i, respectively.

Based on the information provided in Fig. 3, we can define a cost function related to the geometric relationship between the coordinates of APs and distance estimates as follows:

$$J^{geo} = \sum_{i=1}^{T} \sum_{n=1}^{N} w_n(i) \left(||\hat{\mathbf{z}}(i) - \mathbf{z}_n(i)|| - \hat{d}_n(i) \right)^2, \quad (16)$$

where $w_n(i)$ is the weight that compensates for different behaviors of distance measurement errors. There are many options, for instance, $w_n(i) = 1$, $w_n(i) = 1/\hat{d}_n^2(i)$ or $w_n(i) = 1/(\hat{d}_n(i)\hat{s}_n(i)^2)$ as similar to the WLS method. It is obvious that the geometric cost function becomes 0 if the raining module produces the perfect distance estimate.

When the location estimation interval is short (e.g., 500 ms or 1 s), the state of the device hardly changes. To penalize if the estimated position and velocity of the device are inconsistent at consecutive time steps, we can define the following cost functions:

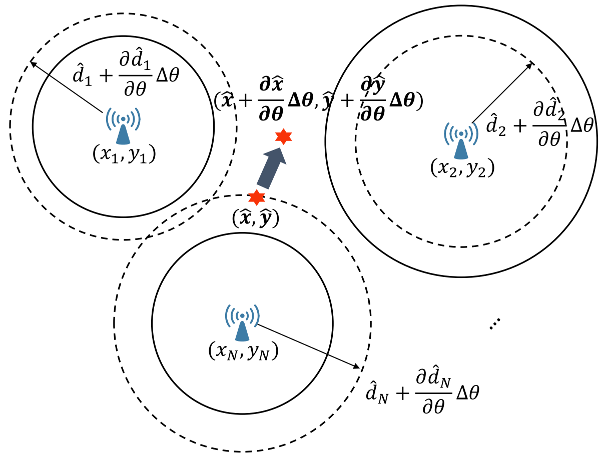
$$\begin{aligned} J^{pos} &= \sum_{i=2}^{T} ||\hat{\mathbf{z}}(i) - \hat{\mathbf{z}}(i-1)||^2, \\ J^{velo} &= \sum_{i=3}^{T} ||\hat{\mathbf{v}}(i) - \hat{\mathbf{v}}(i-1)||^2, \end{aligned} \quad (17)$$

where $\hat{\mathbf{v}}(i) = (\hat{\mathbf{z}}(i) - \hat{\mathbf{z}}(i-1))/\Delta T$ is the estimated velocity with a positioning interval of ΔT.

One last option is to use differently measured distances. This cost function is designed to exploit the distance measurement obtained with the FTM protocol, but it is applicable to other distance measurements. If we have relatively accurate distance measurements from the APs, we can use them as the guide. The cost function related to the guide is defined by

$$J^{guide} = \sum_{i=1}^{T} \sum_{n=1}^{N} I_n(i) \left(\hat{d}_n(i) - d_n^{guide}(i) \right)^2, \quad (18)$$

where $I_n(i) = 1$ if the guide from the n-th AP at time step i, denoted by $d_n^{guide}(i)$, is available and 0 otherwise.



$\hat{d}_1 + \frac{\partial \hat{d}_1}{\partial \theta}\Delta\theta$
(x_1, y_1)
$(\hat{x} + \frac{\partial \hat{x}}{\partial \theta}\Delta\theta, \hat{y} + \frac{\partial \hat{y}}{\partial \theta}\Delta\theta)$
$\hat{d}_2 + \frac{\partial \hat{d}_2}{\partial \theta}\Delta\theta$
(x_2, y_2)
$(\hat{x}, \hat{y})$
(x_N, y_N)
$\hat{d}_N + \frac{\partial \hat{d}_N}{\partial \theta}\Delta\theta$

FIGURE 4. Visualization of the gradient computation. If a parameter θ changes by $\Delta\theta$, it simultaneously affects the distance estimate from every anchor node. As a result, the estimated coordinates of the device and the geometric cost will be changed.

C. COST FUNCTIONS FOR SEMI-SUPERVISED LEARNING

System operators may collect some true coordinates using their own devices for preparing the training data or verifying the positioning performance. In this case, we can include some ground truth data in the training phase by directly comparing the estimated coordinates with true coordinates. The cost function for this is defined by

$$J^{loc*} = \sum_{i=1}^{T} I(i)||\hat{\mathbf{z}}(i) - \mathbf{z}^*(i)||^2, \tag{19}$$

where $I(i) = 1$ if the true coordinates at time step i, denoted by $\mathbf{z}^*(i)$, are measured and 0 otherwise. Using the measured true coordinates of the device, we can also know the true distances from neighboring APs. The cost function related to the distance estimation is given by

$$J^{dist*} = \sum_{i=1}^{T}\sum_{n=1}^{N} I(i)\left(||\hat{\mathbf{z}}_n(i) - \mathbf{z}^*(i)|| - \hat{d}_n(i)\right)^2. \tag{20}$$

Note that the above two cost functions will become 0 in case no ground truth coordinates are measured.

D. TRAINING PHASE

By combining all the cost functions introduced in this section, we can obtain a unified cost function as follows:

$$J = \lambda_1 J^{geo} + \lambda_2 J^{pos} + \lambda_3 J^{velo} + \lambda_4 J^{guide} + \lambda_5 J^{loc*} + \lambda_6 J^{dist*}, \tag{21}$$

where all $\lambda \geq 0$ in the equation are constants that control the balance between cost functions.

In the training phase, we iteratively update every trainable parameter in the direction of minimizing the unified cost function. For instance, a simple gradient descent method updates any parameter $\theta \in \tilde{\Theta}$ as

$$\hat{\theta} \leftarrow \hat{\theta} - \alpha\frac{\partial J}{\partial \theta}, \tag{22}$$

where α is the learning rate and $\hat{\theta}$ represents the estimate of θ. Using the chain rule, the above derivative is expressed by

$$\frac{\partial J}{\partial \theta} = \sum_{i=1}^{T}\sum_{n=1}^{N} \frac{\partial J}{\partial \mathbf{m}_n(i)}\frac{\mathbf{m}_n(i)}{\partial \theta}, \tag{23}$$

where $\mathbf{m}_n(i)$ is the vector defined in equation (10) with respect to the n-th anchor node at time step i. As J is differentiable with respect to $\hat{\mathbf{z}}(i)$ for $i = 1, \ldots, T$ and $\hat{\mathbf{z}}(i)$ is again differentiable with respect to $\mathbf{m}_n(i)$ for $n = 1, \ldots, N$, we can obtain the derivative in equation (23).

Fig. 4 visualizes the meaning of equation (23). If θ changes slightly by $\Delta\theta$, it will simultaneously affect the distance estimate (and standard deviation estimate as well) for every anchor node. Some circles will expand and some will shrink depending on the input, and as a result, the estimated coordinates of the device will change. Finally, the unified cost function will change by ΔJ, and we can obtain equation (23). In the implementation phase, we use the back propagation algorithm to compute the gradient [58].

E. CONVERGENCE ISSUE

Because NNs have a number of parameters, they are able to learn flexible ways of producing output data from input data, but they also have high chances of experiencing an over fitting problem. If there are only three APs in the area of interest and we train an NN-based ranging module in an unsupervised learning manner using the cost functions introduced in this section, the module will always produce the same distance output regardless of the input so that the estimated coordinates of the device have the same distance from the three APs (this point is called the circumcenter of a triangle). In this case, the unified cost function will be 0.

However, if the number of APs increases and we include more than three APs in the positioning phase, the opportunity that a single distance will result in 0 unified cost is greatly reduced. In this case, the NNs try to produce different outputs for different inputs. In addition, we can exploit a different set of APs for different time steps. In this case, the cost functions in equation (17) will prevent the NNs for producing trivial outputs. This is because, to minimize these cost functions, the estimated coordinates (or velocity) of the device must be similar between two consecutive time steps regardless of the choice of anchor nodes.

V. PERFORMANCE EVALUATION

A. EXPERIMENT SETUP

Fig. 5 shows the floor plan of the experiment site, which is a typical indoor office environment. The total area is approximately 8,600 m^2 and 59 IEEE 802.11ac APs are installed in the ceiling to cover the entire area. Each AP broadcasts a beacon signal on both 2.4 and 5 GHz bands. The APs use one out of three non-overlapping channels for the 2.4 GHz band (i.e., Wi-Fi channels 1, 6, and 11) and one out of 24 channels for the 5 GHz band. By receiving a beacon signal, each device

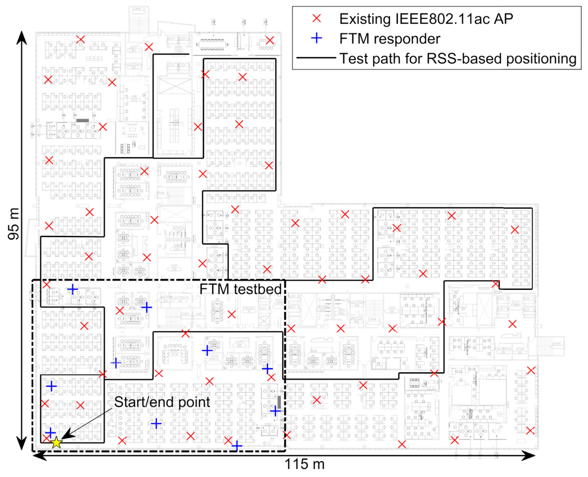


FIGURE 5. **Floor plan of the experiment site. The location of existing APs and newly installed 10 FTM responders are presented.**

can measure the RSS and use the measured value to estimate distance.

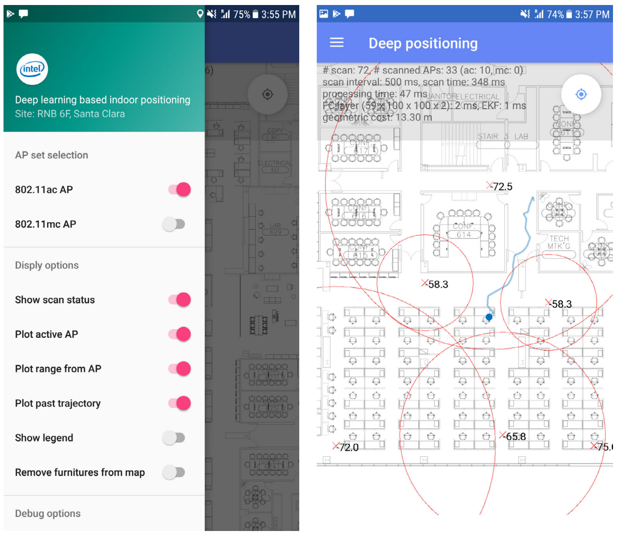
In addition to the existing APs, we installed 10 IEEE 802.11-2016 standard capable APs in a 56 × 37 m^2 area to verify the performance of RTT-based positioning. These APs were installed on top of cubicles or in conference rooms. The newly installed APs are small computers equipped with an Intel Atom CPU and Intel AC8260 Wi-Fi chipset that supports FTM functionality. These APs work in the FTM responder (FTMR) mode, which responds to the RTT measurement request from the devices. Therefore, we use the term FTMR to refer to these APs.

Each FTMR operates with the 40 MHz bandwidth configuration on the 5 GHz band using Wi-Fi channel 40 or 48. By default, each FTMR and each device performs ranging procedures eight times for a single RTT measurement request. Subsequently, the device can obtain the average distance estimate, the standard deviation, and RSS. While serving as an FTMR, the AP also broadcasts a beacon on the same channel that FTM operates on. Therefore, devices that do not support RTT features may use FTMRs as anchor nodes for RSS-based positioning.

We used Android devices to obtain RSS or RTT measurements from APs in the vicinity. We have tested many devices running on various Android versions. The list of devices is summarized in Table 2. Because the RTT feature was newly added to Android version 9, devices with Android version 8 or below were used to verify the performance of RSS-based positioning. In addition, after testing various devices, we noticed that only the Google Pixel series supports RTT features at this moment. Therefore, Pixel series was used to verify the performance of RTT-based positioning.

B. REAL-TIME LOCATION TRACKING APPLICATION

We have developed a real-time location tracking application for Android devices to collect training data and to verify the proposed method. Fig. 6 shows the screenshots of the application. Depending on the availability of the RTT feature,



Deep learning based indoor positioning
Site: RNB 6F, Santa Clara

AP set selection

802.11ac AP [✓]

802.11mc AP

Disply options

Show scan status [✓]

Plot active AP [✓]

Plot range from AP [✓]

Plot past trajectory [✓]

Show legend

Remove furnitures from map

Debug options

Deep positioning

scan: 72, # scanned APs: 33 (ac: 10, mc: 0)
scan interval: 500 ms, scan time: 348 ms
processing time: 47 ms
FC layer (59 x 100 x 100 x 2): 2 ms, EKF: 1 ms
geometric cost: 13.30 m

FIGURE 6. **Real-time location tracking application. Depending on the Android version, the application collects RSS or RTT measurements for positioning (a demo video is available online).**

each device collects RSS or RTT measurements every 500 ms and feeds these measurements to the ranging module. Once distances and standard deviations are calculated, the application estimates the coordinates of the device using the EKF method with up to five of the closest APs.

To measure the RSS, the Android API (application program interface) provides a Wi-Fi scan function, which is the startScan method under the WiFiManager class. This method actually invokes another startScan method implemented in a non-public class called WiFiServiceImpl. An issue is that the provided public startScan method scans all available channels in both 2.4 and 5 GHz bands, and therefore, scan results are updated every 3-5 s, which is not enough to track the moving users.

To improve the scan speed, we implemented a customized scan method that scans only selected channels. We specified channel numbers in an instance of the ScanSettings class and invoked the startScan method by passing this instance. Using this method, we could quickly obtain scan results; for instance, it takes less than 500 ms for almost every device to scan channels 1, 6, and 11 only.

RTT-related measurements using the FTM protocol can be obtained through another class called WifiRttManager. First, the device scans all the neighboring APs using the StartScan method in the WifiManager class and prepares a list of FTMRs. Then, the device periodically sends distance measurement requests to nearby FTMRs.

To implement the NNs, we used Tensorflow lite (version 1.12), which can run a pre-trained NN on mobile devices. Therefore, we trained NNs on a server using all the training data obtained from multiple devices and exported the trained model for the devices. Note that all the computation tasks, including Wi-Fi scanning, ranging, and positioning run on an additional thread, whereas the main thread is only involved in updating the location of the device on screen.

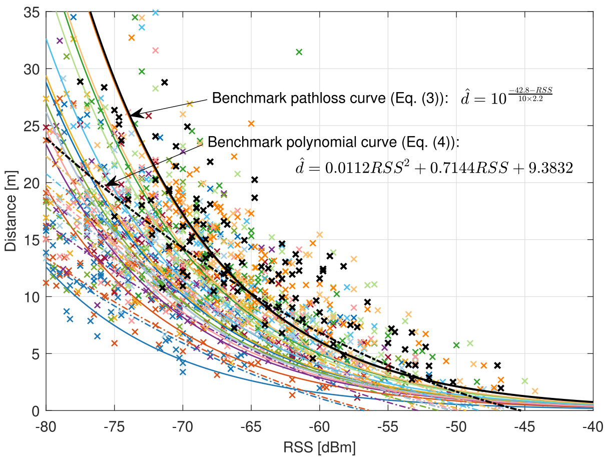


FIGURE 7. Relationship between RSS and true distance obtained from the test data. Data measured in the device having the smallest offset are presented with black color.

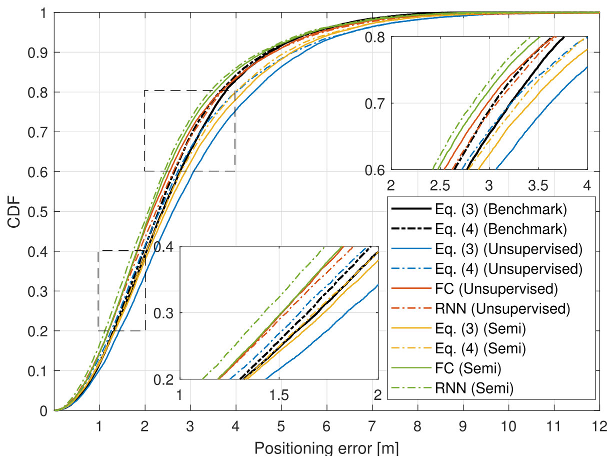


FIGURE 8. CDF of RSS-based positioning accuracy.

TABLE 3. Summary of RSS-based positioning results.

Learning type	Positioning method	MAE [m]	RMSE [m]	90%-tile [m]
Benchmark [a]	Eq. (3) + LS	5.583	7.410	10.368
	Eq. (4) + LS	5.051	6.762	9.130
	Eq. (3) + WLS	4.681	6.207	8.292
	Eq. (4) + WLS	4.578	5.744	8.313
	Eq. (3) + EKF	**2.648**	**3.063**	**4.740**
	Eq. (4) + EKF	**2.584**	**3.012**	**4.689**
Unsupervised	**Eq. (3) + EKF**	**2.988**	**3.504**	**5.488**
	Eq. (4) + EKF	**2.744**	**3.308**	**5.252**
	FC + EKF	2.542	3.055	4.838
	RNN + EKF	2.570	3.064	4.804
Semi-supervised [b]	**Eq. (3) + EKF**	**2.816**	**3.313**	**5.233**
	Eq. (4) + EKF	**2.750**	**3.246**	**5.086**
	FC + EKF	2.463	2.919	4.608
	RNN + EKF	2.397	2.876	4.540

[a]Parameters and offsets are perfectly optimized on the test data.
[b]LG G6 collected 115 true coordinates (0.4% of total training data).

C. EXPERIMENTS WITH EXISTING Wi-Fi APs

To obtain RSS measurements, we scanned only three channels on the 2.4 GHz band. For the ranging module, we used equations (3) and (4), and simple NN structures: (a) Two FC layers with (100, 100) hidden nodes and (b) a single-layer LSTM-RNN with 100 hidden nodes. Each output of the NNs is limited to $d_{max} = 100$ m and $s_{max} = 50$ m.

We collected training data by randomly walking around the indoor area for 15 min with each device. To mimic a system operator collecting ground truth data, an LG G6 device was used to collect 115 true coordinates at certain landmark positions (e.g., center of room, near the pole, and so on). The amount of collected ground truth data was only 0.4% of the total amount of training data (i.e., 27,000 = 15 devices × 15 min / 500 ms). The test data were acquired by walking along the test path shown in Fig. 5. The length of the test path was 580 m, and it took another 10 min for each device. We measured time when the devices passed through each edge of the test path to get the true coordinates at all time steps using interpolation. Because data collection was done during the daytime, human effects such as body shadowing are included in both training and test data.

For the benchmark purposes, we chose optimal parameters for the model-based ranging module. Fig. 7 shows the relationship between the RSS and true distance obtained from the test data that have true coordinates of devices. The parameters in the ranging module were chosen to minimize the normalized mean squared error (NMSE), which is defined as $NMSE = E[((\hat{d} - d^*)/d^*)^2]$ with the true distance d^*. Meanwhile, we also selected a proper offset for each device, where the minimum offset was assumed to be 0 dB.

The bold solid line in Fig. 7 represents the benchmark pathloss curve with 0 dB offset. The optimal parameters were chosen as $RSS(d_0) = -42.8$ and $\eta = 2.2$. Similarly, the bold dotted line represents the benchmark polynomial curve, where $c_2 = 0.0112$, $c_1 = 0.7144$, and $c_0 = 9.3832$. In addition, we chose an optimal proportional coefficient β in equation (5); for instance, 0.05 and 0.04 were selected for the pathloss and polynomial-based methods with the EKF. *By doing so, the benchmark scenarios perfectly optimized all the trainable variables on the test data, and thus, they produce the best performance for the test data.*

On the other hand, the proposed method optimizes every trainable parameter using the collected training data. We use 70% of the training data to update parameters and 30% of the data to validate the parameter update at each epoch. The Adam optimizer is used with the learning rate of 10^{-3} [59]. We assume $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6) = (1, 0.1, 0.1, 0, 0, 0)$ for unsupervised learning scenarios and $(1, 0.1, 0.1, 0, 1, 1)$ for semi-supervised learning scenarios.

Table 3 summarizes the performance of each scenario. The main performance metrics are the mean absolute error (MAE) and the root mean squared error (RMSE) with respect to the true coordinate $\mathbf{z}^*$, which are defined by $MAE = E[||\hat{\mathbf{z}} - \mathbf{z}^*||]$ and $RMSE = \sqrt{E[||\hat{\mathbf{z}} - \mathbf{z}^*||^2]}$, respectively. In addition, the 90-th percentile accuracy is presented in the table. For

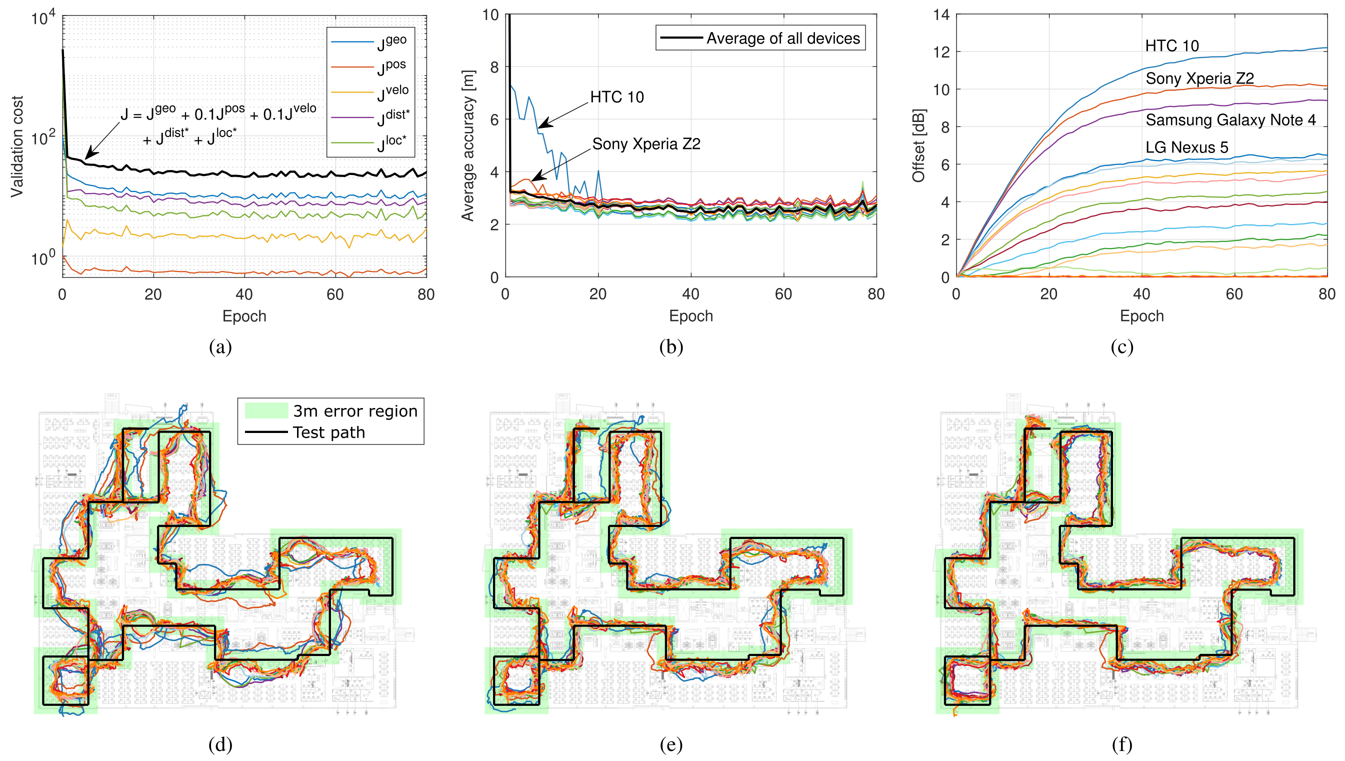


FIGURE 9. **Training details: (a) Costs with respect to the validation data, (b) per-device average positioning accuracy on the test data, (c) trained offset of each device (name of top four devices are presented), (d) estimated test trajectories for all devices after one training epoch, (e) after 10 epochs, and (f) after 50 epochs.**

every case, we consider the offset of the device only because the existing APs are the same model.

For the benchmark scenarios, we verified the performances of all the positioning methods discussed in Section III-D. As the table shows, the EKF method outperforms the LS or WLS methods because it estimates the coordinates using previous estimates. Therefore, we mainly use the EKF method for the remainder of the paper.

In cases where model-based ranging modules are used, the positioning accuracy of unsupervised learning scenarios is close to the benchmark accuracy; for instance, the performances with equation (3) and (4) are 87% and 94% of their counterpart benchmark performances, respectively, in terms of MAE. If 0.4% of the ground truth coordinates are used in the training phase for semi-supervised learning, both cases achieve 94% of the benchmark performances. If NNs are adopted in the ranging module, the obtained positioning accuracy is higher than the best model-based benchmark accuracy obtained using equation (4) even though the module is trained in an unsupervised learning way. Semi-supervised learning scenarios with NNs further improve the positioning accuracy. The same results can be observed in Fig. 8, which shows the cumulative density function (CDF) of the positioning errors for all devices.

Fig. 9 shows the details of the training phase. In this figure, a semi-supervised learning scenario with the RNN is used. The validation cost tends to decrease with the training epoch and the average positioning accuracy decreases accordingly. Devices having offsets that are too far from the

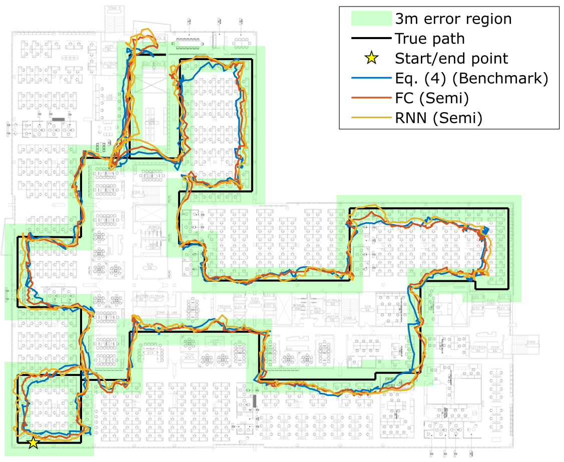


FIGURE 10. **Estimated test trajectory using samsung galaxy note 5.**

average (e.g., Sony Xperia Z2 or HTC 10) initially produce inaccurate positioning results compared to others, but soon achieve similar accuracy levels to others as their offsets are optimized appropriately. Fig. 9(d), 9(e), and 9(f) represent the estimated test trajectories for all devices after 1, 10, and 50 training epochs, respectively. After 10 training epochs, the estimated trajectories are closer to the true path, except for some devices. After a sufficient number of training epochs, all the estimated trajectories are close to the true path. The area with a green color represents the 3 m error region, in which coordinates are less than 3 m apart from the closest test path.

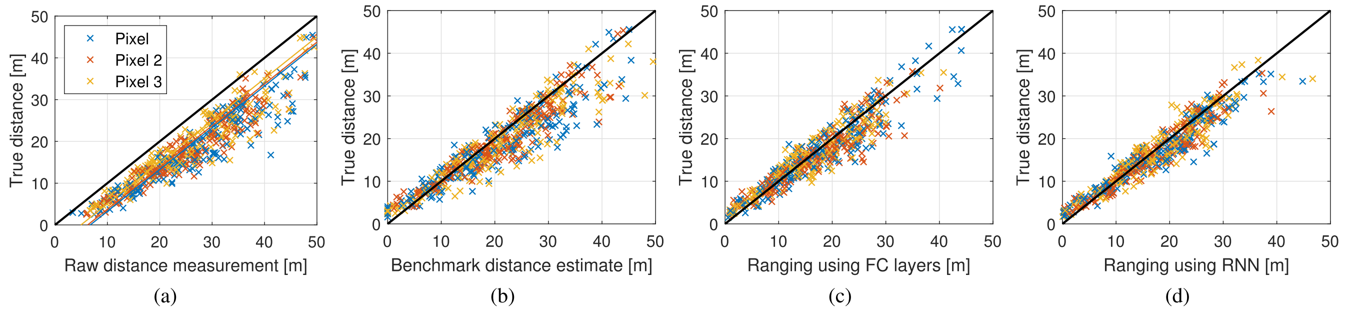


FIGURE 11. Relationship between true distance and (a) raw distance measurement from the FTM protocol, (b) benchmark distance estimate, (c), distance estimate using the FC layers, and (d) distance estimate using the RNN.

Finally, Fig. 10 illustrates the estimated test trajectories for the selected scenarios using Samsung Galaxy Note 5, which was not involved in collecting ground truth coordinates. The estimated trajectories for all scenarios are in the 3 m error region most of the time.

D. EXPERIMENTS UNDER THE FTM TESTBED

We verified the performance of the RTT-based positioning method with 10 FTMRs and Google Pixel series. Similar to the previous experiments, we measured distances from nearby FTMRs every 500 ms. We collected training data by moving around the FTM testbed area for 10 min with each device. In addition, the test data were collected by following the pre-defined test path with a length of 235 m. It took approximately 4 min to obtain the test data for each device. We used the same NN structures and optimizer as in the previous experiments.

For the purpose of performance comparison, we also evaluated the benchmark performance. To this end, we chose the distance measurement offset in equation (6) to minimize the NMSE of distance estimation using the test data. However, the proposed method optimizes every trainable parameter using the training data. In this experiment, we did not measure the true coordinates for semi-supervised learning purposes, because the FTM protocol produces quite reliable distance estimates, which can be used as guides. Therefore, we used $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6) = (1, 0.1, 0.1, 1, 0, 0)$ for training.

Fig. 11(a) shows that the raw distance measurements using the FTM protocol have negative offsets from the true distances. For cases in which equation (6) is used to compensate for the offsets, the points are shifted vertically toward the perfect line (i.e., black solid line) as shown in Fig. 11(b). In addition, NN-based ranging modules trained in an unsupervised learning way also produce accurate distance estimates. Fig. 11(c) and 11(d) show the ranging results using the FC layers and RNN, respectively, where points look more concentrated around the perfect line than in Fig. 11(b).

The CDF of the ranging error is depicted in Fig. 12. As we mentioned previously, the raw distance measurements obtained with the FTM protocol contain significant errors. The benchmark and unsupervised learning results obtained using equation (6) produced overlapping CDF curves because the offsets estimated using these two approaches were almost the same, as summarized in Table 4. The figure shows that the NN-based ranging modules outperforms the model-based benchmark performances. This is because the NNs are able to learn non-linear calibration curves instead of simply optimizing a distance measurement offset for each device. In addition, the NNs produce outputs from not only raw distance measurements, but also the standard deviation and RSS reported by the FTM protocol. Therefore, they can learn how to efficiently combine these information during the training stage. For instance, if the raw distance report is short while RSS is low, the NNs may infer the propagation condition as an NLOS condition.

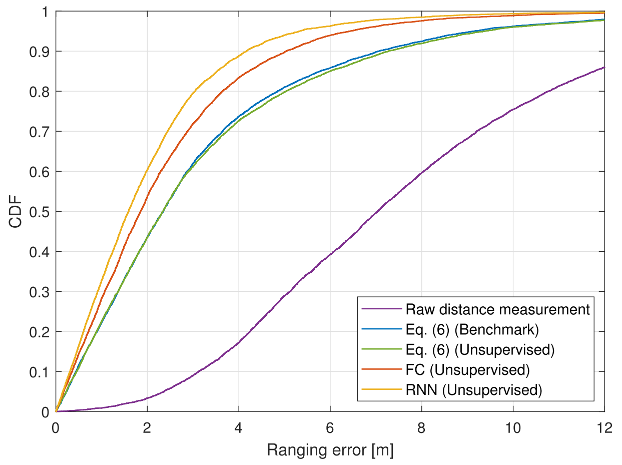


FIGURE 12. CDF of RTT-based ranging accuracy.

The performance of RTT-based positioning per device is summarized in Table 4. If a model-based ranging module is used, the only trainable variable is the offset of each device. The table shows that the offsets in equation (6) determined by unsupervised learning for each device are considerably close to the offset values that are perfectly optimized for the test data. For this reason, the performance of unsupervised learning is close to the benchmark performance. An interesting observation is that offsets optimized with the NNs have relative values between the devices, and the absolute offsets are processed inside the NNs. In addition, the offsets for the NNs have the same dimension as the input layer. Therefore, each device has its own standard deviation, RSS, and distance offsets.

TABLE 4. Summary of RTT-based positioning result per device (40 MHz).

Positioning method [a]	Device number	Offsets [b] [m]	Offsets [b] [dB]	MAE [m]	RMSE [m]	90%-tile [m]
Benchmark	16	-6.37	-	2.066	2.433	4.030
	17	-5.83	-	2.240	2.733	4.224
	18	-4.66	-	2.097	2.413	3.804
Eq. (6)	16	-6.64	-	2.000	2.345	3.841
	17	-5.97	-	2.177	2.648	4.105
	18	-4.65	-	2.101	2.419	3.821
FC	16	0	0.19	1.637	1.927	3.075
	17	1.56	0	1.459	1.735	2.932
	18	2.35	3.00	1.545	1.784	2.869
RNN	16	0	0.07	1.809	2.042	3.204
	17	1.43	0	1.597	1.810	2.847
	18	2.11	2.66	1.685	1.890	2.918

[a]EKF is used for all scenarios.

[b]Both distance and RSS offsets were optimized when NNs are used.

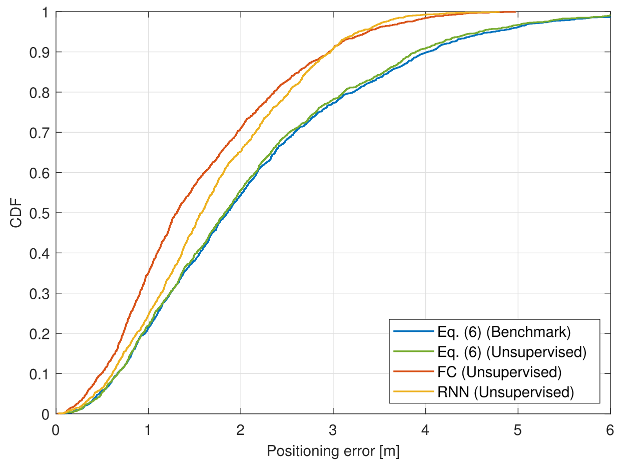


FIGURE 13. CDF of RTT-based positioning accuracy.

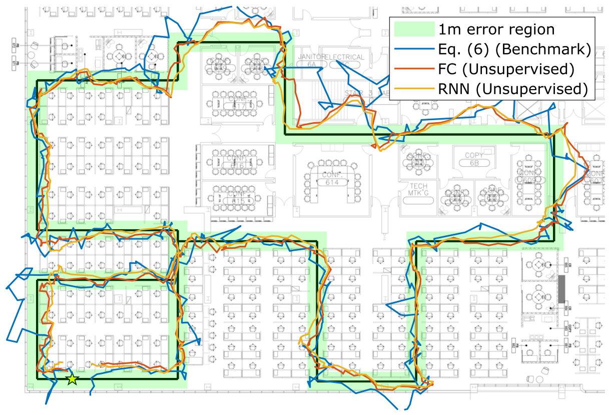


FIGURE 14. Estimated test trajectory using Google pixel 2.

Fig. 13 depicts the CDF of RTT-based positioning accuracy for selected scenarios. As we have discussed so far, the unsupervised learning approach is able to produce almost the same positioning error as the benchmark. As NN-based ranging modules provide more accurate distance estimates than model-based ranging module, the positioning accuracy was improved with NNs. Finally, Fig. 14 illustrates the estimated test trajectories using Google Pixel 2. Because we can obtain relatively accurate positioning results using the FTM protocol, the trajectories are in the 1 m error region most of the time. Other Pixel devices have similar trajectories.

VI. CONCLUSION

In this paper, we studied unsupervised learning techniques to adaptively adjust trainable parameters in the ranging module depending on the surrounding environments. The main intuition of unsupervised learning is that the accuracy of the ranging module can be inferred from the geometry. Based on this intuition, the cost functions were designed to optimize parameters in a way that yielded more reasonable results. In addition, the proposed method was able to compensate for each device's unique characteristics by simply assigning an offset to each device. As errors propagated in the backward direction through the positioning and ranging module to the input layer, every parameter including the offset of each device was iteratively optimized. The results of the experiments revealed that the proposed method closely achieves the benchmark performance with the model-based ranging module. In addition, the positioning accuracy with NN-based ranging modules are superior to the benchmark positioning performance.

APPENDIX

EXTENDED KALMAN FILTER

We briefly summarize the EKF procedure used in this work. The unknown state is assumed to be the coordinates of the device. Once new measurements are available, the EKF first predicts the state based on the previous state estimate and corrects the predicted state by using these measurements.

The state transition model for the prediction is given by

$$\mathbf{z}(i) = \mathbf{z}(i-1) + \mathbf{v}(i)\Delta T, \quad i \geq 1, \tag{24}$$

where $\mathbf{v}(i)$ represents the average velocity vector between time steps $i-1$ and i, and ΔT denotes the measurement interval. We assume that $\mathbf{v}(i)$ is a random vector with zero mean and the covariance matrix of $\mathbf{Q}(i) = (\Delta T)^2 \; E[\mathbf{v}(i)\mathbf{v}(i)^T]$. Therefore, we have the following equations:

$$\begin{aligned}\hat{\mathbf{z}}(i|i-1) &= \hat{\mathbf{z}}(i-1|i-1),\\ \mathbf{P}(i|i-1) &= \mathbf{P}(i-1|i-1) + \mathbf{Q}(i),\end{aligned} \tag{25}$$

where $\hat{\mathbf{z}}(i_1|i_2)$ represents the predicted state at time step i_1 using all measurements up to time step i_2, and $\mathbf{P}(i_1|i_2)$ is its covariance matrix. The initial coordinates $\hat{\mathbf{z}}(0|0)$ are simply assumed as the center of neighboring anchor nodes.

The measurement of the system are expressed as the vector of distance estimates from N anchor nodes as follows:

$$\hat{\mathbf{d}}(i) = \mathbf{h}(\mathbf{z}(i), \mathbf{w}(i)) = \begin{bmatrix} ||\mathbf{z}(i) - \mathbf{z}_1(i)|| \\ \vdots \\ ||\mathbf{z}(i) - \mathbf{z}_N(i)|| \end{bmatrix} + \mathbf{w}(i), \tag{26}$$

where $\hat{\mathbf{d}}(i) = [\hat{d}_1(i), \ldots, \hat{d}_N(i)]^T$ is the measurement vector, $\mathbf{w}(i)$ is the measurement noise whose covariance matrix is given by $\mathbf{R}(i) = E[\mathbf{w}(i)\mathbf{w}(i)^T] = \mathrm{diag}(\hat{s}_1(i)^2, \ldots, \hat{s}_N(i)^2)$.

Using this measurement model, we can predict the distance measurements at time step i as $\mathbf{h}(\hat{\mathbf{z}}(i|i-1), \mathbf{0}_N)$, and the gap between this prediction and the actual measurement is called innovation. The innovation and its covariance matrix are represented by

$$\begin{aligned}\mathbf{e}(i) &= \hat{\mathbf{d}}(i) - \mathbf{h}(\hat{\mathbf{z}}(i|i-1), \mathbf{0}_N),\\ \mathbf{S}(i) &= \mathbf{H}(i)^T\mathbf{P}(i|i-1)\mathbf{H}(i) + \mathbf{R}(i),\end{aligned} \tag{27}$$

where $\mathbf{H}(i) \in \mathbb{R}^{N\times 2}$ is the Jacobian matrix defined by

$$\mathbf{H}(i) = \frac{\partial \mathbf{h}(\mathbf{z}(i), \mathbf{0}_N)}{\partial \mathbf{z}(i)}. \tag{28}$$

The above derivative is evaluated at $\mathbf{z}(i) = \hat{\mathbf{z}}(i|i-1)$.

Finally, the Kalman gain, updated state, and covariance matrix of the state at time step i are respectively given by

$$\begin{aligned}\mathbf{K}(i) &= \mathbf{P}(i|i-1)\mathbf{H}(i)^T\mathbf{S}(i)^{-1},\\ \hat{\mathbf{z}}(i|i) &= \hat{\mathbf{z}}(i|i-1) + \mathbf{K}(i)\mathbf{e}(i),\\ \mathbf{P}(i|i) &= (\mathbf{I}_2 - \mathbf{K}(i)\mathbf{H}(i))\mathbf{P}(i|i-1).\end{aligned} \tag{29}$$

Note that $\hat{\mathbf{z}}(i|i)$ indicates the estimated coordinates at time step i using every measurement, and thus, $\hat{\mathbf{z}}^{EKF}(i) = \hat{\mathbf{z}}(i|i)$.

ACKNOWLEDGMENT

The authors would like to thank Doyoung Ham from Seoul National University (e-mail: wj2dy@maxwell.snu.ac.kr) for helping us to implement the customized RSS scan.

REFERENCES

[1] J. Medbo, I. Siomina, A. Kangas, and J. Furuskog, "Propagation channel impact on LTE positioning accuracy: A study based on real measurements of observed time difference of arrival," in *Proc. IEEE 20th Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2009, pp. 2213–2217.

[2] A. Dammann, R. Raulefs, and S. Zhang, "On prospects of positioning in 5G," in *Proc. IEEE Int. Conf. Commun. Workshops (ICCW)*, Piscataway, NJ, USA, Jun. 2015, pp. 1207–1213.

[3] J. A. del Peral-Rosado, R. Raulefs, J. A. López-Salcedo, and G. Seco-Granados, "Survey of cellular mobile radio localization methods: From 1G to 5G," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1124–1148, 2nd Quart., 2017.

[4] R. Mendrzik, H. Wymeersch, G. Bauch, and Z. Abu-Shaban, "Harnessing NLOS components for position and orientation estimation in 5G millimeter wave MIMO," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 93–107, Jan. 2019.

[5] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. 19th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 2, Mar. 2000, pp. 775–784.

[6] P. Prasithsangaree, P. Krishnamurthy, and P. K. Chrysanthis, "On indoor position location with wireless LANs," in *Proc. 13th IEEE Int. Symp. Personal, Indoor, Mobile Radio Commun.*, vol. 2. Sep. 2002, pp. 720–724.

[7] Y.-C. Wang, X. Jia, and H. K. Lee, "An indoors wireless positioning system based on wireless local area network infrastructure," in *Proc. 6th Int. Symp. Satell. Navigat. Technol. Including Mobile Positioning Location Services*, 2003, pp. 1–13.

[8] T. Kitasuka, K. Hisazumi, T. Nakanishi, and A. Fukuda, "WiPS: Location and motion sensing technique of IEEE 802.11 devices," in *Proc. 3rd Int. Conf. Inf. Technol. Appl.*, vol. 2, Jul. 2005, pp. 346–349.

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

[10] J. Yang and Y. Chen, "Indoor localization using improved RSS-based lateration methods," in *Proc. IEEE Global Telecommun. Conf.*, Nov./Dec. 2009, pp. 1–6.

[11] B. Kim, W. Bong, and Y. C. Kim, "Indoor localization for Wi-Fi devices by cross-monitoring AP and weighted triangulation," in *Proc. IEEE Consum. Commun. Netw. Conf.*, Jan. 2011, pp. 933–936.

[12] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[13] L. Banin, U. Schatzberg, and Y. Amizur, "WiFi FTM and map information fusion for accurate positioning," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, Oct. 2016, pp. 1–4.

[14] L. Banin, O. Bar-Shalom, N. Dvorecki, and Y. Amizur, "High-accuracy indoor geolocation using collaborative time of arrival (CToA)," Intel White Paper, Santa Clara, CA, USA, Sep. 2017.

[15] M. Ibrahim, H. Liu, M. Jawahar, V. Nguyen, M. Gruteser, R. Howard, B. Yu, and F. Bai, "Verification: Accuracy evaluation of WiFi fine time measurements on an open platform," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, Nov. 2018, pp. 417–427.

[16] N. Dvorecki, O. Bar-Shalom, L. Banin, and Y. Amizur, "A machine learning approach for Wi-Fi RTT ranging," in *Proc. Int. Tech. Meeting Inst. Navigat.*, Jan. 2019, pp. 435–444.

[17] L. Zhang, X. Liu, J. Song, C. Gurrin, and Z. Zhu, "A comprehensive study of Bluetooth fingerprinting-based algorithms for localization," in *Proc. 27th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Mar. 2013, pp. 300–305.

[18] Y. Wang, X. Yang, Y. Zhao, Y. Liu, and L. Cuthbert, "Bluetooth positioning using RSSI and triangulation methods," in *Proc. IEEE 10th Consum. Commun. Netw. Conf.*, Jan. 2013, pp. 837–842.

[19] J. Röbesaat, P. Zhang, M. Abdelaal, and O. Theel, "An improved BLE indoor localization with Kalman-based fusion: An experimental study," *Sensors*, vol. 17, no. 5, p. 951, 2017.

[20] J. Khodjaev, Y. Park, and A. S. Malik, "Survey of NLOS identification and error mitigation problems in UWB-based positioning algorithms for dense environments," *Ann. Telecommun.*, vol. 65, nos. 5–6, pp. 301–311, 2010.

[21] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. Al-Ammar, and H. Al-Khalifa, "Ultra wideband indoor positioning technologies: Analysis and recent advances," *Sensors*, vol. 16, no. 5, p. 707, 2016.

[22] G. Schroeer, "A real-time UWB multi-channel indoor positioning system for industrial scenarios," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, Sep. 2018, pp. 1–5.

[23] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "LANDMARC: Indoor location sensing using active RFID," *Wireless Netw.*, vol. 10, no. 6, pp. 701–710, 2004.

[24] B. Ozdenizci, K. Ok, V. Coskun, and M. N. Aydin, "Development of an indoor navigation system using NFC technology," in *Proc. 4th Int. Conf. Inf. Comput.*, Apr. 2011, pp. 11–14.

[25] W. Kang and Y. Han, "SmartPDR: Smartphone-based pedestrian dead reckoning for indoor localization," *IEEE Sensors J.*, vol. 15, no. 5, pp. 2906–2916, May 2015.

[26] E. M. Diaz, "Inertial pocket navigation system: Unaided 3D positioning," *Sensors*, vol. 15, no. 4, pp. 9156–9178, 2015.

[27] B. Zhou, Q. Li, Q. Mao, W. Tu, and X. Zhang, "Activity sequence-based indoor pedestrian localization using smartphones," *IEEE Trans. Human-Mach. Syst.*, vol. 45, no. 5, pp. 562–574, Oct. 2015.

[28] H. Ju, S. Y. Park, and C. G. Park, "A smartphone-based pedestrian dead reckoning system with multiple virtual tracking for indoor navigation," *IEEE Sensors J.*, vol. 18, no. 16, pp. 6756–6764, Aug. 2018.

[29] A. Solin, S. Cortes, E. Rahtu, and J. Kannala, "Inertial odometry on handheld smartphones," in *Proc. 21st Int. Conf. Inf. Fusion*, Jul. 2018, pp. 1–5.

[30] W. Elloumi, K. Guissous, A. Chetouani, R. Canals, R. Leconge, B. Emile, and S. Treuillet, "Indoor navigation assistance with a Smartphone camera based on vanishing points," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, Oct. 2013, pp. 1–9.

[31] W. Elloumi, A. Latoui, R. Canals, A. Chetouani, and S. Treuillet, "Indoor pedestrian localization with a smartphone: A comparison of inertial and vision-based methods," *IEEE Sensors J.*, vol. 16, no. 13, pp. 5376–5388, Jul. 2016.

[32] L. Zheng, W. Zhou, W. Tang, X. Zheng, H. Yang, S. Pu, C. Li, B. Tang, and Y. Chen, "A foot-mounted sensor based 3D indoor positioning approach," in *Proc. IEEE 12th Int. Symp. Auton. Decentralized Syst.*, Mar. 2015, pp. 145–150.

[33] Y. Wang, X. Li, and J. Zou, "A foot-mounted inertial measurement unit (IMU) positioning algorithm based on magnetic constraint," *Sensors*, vol. 18, no. 3, p. 741, 2018.

[34] H. Zhao, Z. Wang, S. Qiu, Y. Shen, L. Zhang, K. Tang, and G. Fortino, "Heading drift reduction for foot-mounted inertial navigation system via multi-sensor fusion and dual-gait analysis," *IEEE Sensors J.*, vol. 19, no. 19, pp. 8514–8521, Oct. 2019.

[35] X. Niu, Y. Li, J. Kuang, and P. Zhang, "Data fusion of dual foot-mounted IMU for pedestrian navigation," *IEEE Sensors J.*, vol. 19, no. 12, pp. 4577–4584, Jun. 2019.

[36] *Spatial Channel Model for Multiple Input Multiple Output (MIMO) Simulations*, document 3GPP TR25.996, Sep. 2012.

[37] *Study on 3D channel model for LTE*, document 3GPP TR36.873, Sep. 2014.

[38] S. Y. Seidel and T. S. Rappaport, "Site-specific propagation prediction for wireless in-building personal communication system design," *IEEE Trans. Veh. Technol.*, vol. 43, no. 4, pp. 879–891, Nov. 1994.

[39] J.-H. Lee, J.-S. Choi, J.-Y. Lee, and S.-C. Kim, "Permittivity effect of building materials on 28 GHz mmWave channel using 3D ray tracing simulation," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.

[40] I. Kashiwagi, T. Taga, and T. Imai, "Time-varying path-shadowing model for indoor populated environments," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 16–28, Jan. 2010.

[41] J.-H. Jung, J. Lee, J.-H. Lee, Y.-H. Kim, and S.-C. Kim, "Ray-tracing-aided modeling of user-shadowing effects in indoor wireless channels," *IEEE Trans. Antennas Propag.*, vol. 62, no. 6, pp. 3412–3416, Jun. 2014.

[42] *IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks–Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11-2016 (Revision IEEE Standard 802.11-2012), Dec. 2016, pp. 1–3534.

[43] [Online]. Available: https://source.android.com/devices/tech/connect/wifi-rtt

[44] X. Li, "RSS-based location estimation with unknown pathloss model," *IEEE Trans. Wireless Commun.*, vol. 5, no. 12, pp. 3626–3633, Dec. 2006.

[45] A. Bel, J. L. Vicario, and G. Seco-Granados, "Localization algorithm with on-line path loss estimation and node selection," *Sensors*, vol. 11, no. 7, pp. 6905–6925, Jul. 2011.

[46] M. R. Gholami, R. M. Vaghefi, and E. G. Ström, "RSS-based sensor localization in the presence of unknown channel parameters," *IEEE Trans. Signal Process.*, vol. 61, no. 15, pp. 3752–3759, Aug. 2013.

[47] Z. Xiao, H. Wen, A. Markham, N. Trigoni, P. Blunsom, and J. Frolik, "Non-line-of-sight identification and mitigation using received signal strength," *IEEE Trans. Wireless Commun.*, vol. 14, no. 3, pp. 1689–1702, Mar. 2015.

[48] H. J. Jo and S. Kim, "Indoor smartphone localization based on LOS and NLOS identification," *Sensors*, vol. 18, no. 11, P. 3987, 2018.

[49] Z. Zhou, Z. Yang, C. Wu, W. Sun, and Y. Liu, "LiFi: Line-of-sight identification with WiFi," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 2688–2696.

[50] Z. Zhou, Z. Yang, C. Wu, L. Shangguan, H. Cai, Y. Liu, and L. M. Ni, "WiFi-based indoor line-of-sight identification," *IEEE Trans. Wireless Commun.*, vol. 14, no. 11, pp. 6125–6136, Nov. 2015.

[51] C. Wu, Z. Yang, Z. Zhou, K. Qian, Y. Liu, and M. Liu, "PhaseU: Real-time LOS identification with WiFi," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2015, pp. 2038–2046.

[52] J.-S. Choi, W.-H. Lee, J.-H. Lee, J.-H. Lee, and S.-C. Kim, "Deep learning based NLOS identification with commodity WLAN devices," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3295–3303, Apr. 2018.

[53] V.-H. Nguyen, M.-T. Nguyen, J. Choi, and Y.-H. Kim, "NLOS identification in WLANs using deep LSTM with CNN features," *Sensors*, vol. 18, no. 11, p. 4057, 2018.

[54] K. W. Cheung, H. C. So, W.-K. Ma, and Y. T. Chan, "Least squares algorithms for time-of-arrival-based mobile location," *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp. 1121–1130, Apr. 2004.

[55] C. Y. Shih and P. J. Marrón, "COLA: Complexity-reduced trilateration approach for 3D localization in wireless sensor networks," in *Proc. 4th Int. Conf. Sensor Technol. Appl.*, Venice, Italy, Jul. 2010, pp. 24–32.

[56] P. Tarrío, A. M. Bernardos, and J. R. Casar, "Weighted least squares techniques for improved received signal strength based localization," *Sensors*, vol. 11, no. 9, pp. 8569–8592, 2011.

[57] K. Bregar and M. Mohorčič, "Improving indoor localization using convolutional neural networks on computationally restricted devices," *IEEE Access*, vol. 6, pp. 17429–17441, 2018.

[58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," in *Neurocomputing: Foundations of Research*, J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, pp. 696–699.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980.* [Online]. Available: https://arxiv.org/abs/1412.6980

JEONGSIK CHOI received the B.S. degree in electrical engineering from the Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 2010, and the M.S. and Ph.D. degrees in electrical engineering from Seoul National University, Seoul, South Korea, in 2012 and 2016, respectively. From 2016 to 2017, he was a Senior Researcher with the Institute of New Media and Communications, Seoul, South Korea. Since 2017, he has been with Intel Labs, Santa Clara, CA, USA. His research interests include wireless propagation channel measurement/modeling, sensor fusion and positioning algorithms, and applications of machine learning techniques to communication systems.

YANG-SEOK CHOI received the B.S. degree from Korea University, Seoul, South Korea, in 1990, the M.S.E.E. degree from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 1992, and the Ph.D. degree from Polytechnic University, Brooklyn, NY, USA, in 2000. From 1992 to 1996, he was with Samsung Electronics, Company Ltd., Suwon, South Korea, where he developed 32-QAM modem for HDTV and QPSK ASIC for DBS. In 2000, he held a summer intern position at AT&T Labs, Research Shannon Lab, Florham Park, NJ, USA. In 2000, he joined National Semiconductor, East Brunswick, NJ, USA, where he was involved in the development of W-CDMA. From 2001 to 2002, he was a Senior Technical Staff Member of AT&T Labs-Research, Middletown, NJ, USA, where he researched on MIMO systems, OFDM systems, and information theory. From 2002 to 2004, he was with ViVATO, Inc., Spokane, WA, USA, working on smart antenna applications to CSMA protocol, and lens and antenna/beam selection techniques. In 2004, he joined Intel Corporation, Hillsboro, OR, USA, where he studied broadband wireless communications systems and led the Standards Team. Since 2013, he has been with Intel Labs, where he studies future wireless communications. He holds 70+ U.S. patents.

SHILPA TALWAR received the M.S. degree in electrical engineering and the Ph.D. degree in applied mathematics from Stanford University, in 1996. She held several senior technical positions at wireless industry involved in a wide-range of projects, including algorithm design for 3G/4G & WLAN chips, satellite communications, GPS, and others. She is currently the Director of wireless multicomm systems and a Senior Principal Architect with the Wireless Communications Laboratory, Intel Labs, where she leads a research team focused on advancements in network architecture and technology innovations for 5G and contributed to the IEEE and 3GPP standard bodies, including 802.16m, LTE-advanced, and 5G NR. She is also coordinating several university collaborations on 5G and leading Intel strategic research alliance on 5G. She has authored over 70+ technical publications. She holds 60+ patents. Her research interests include heterogeneous networks, multi-radio interworking, mmWave communications, advanced MIMO, and full-duplex and interference mitigation techniques. She was the Co-Chair of the ICC 2014 Workshop on 5G Technologies. She has served as a Co-Editor for a Special Issue on the 5G Revolution of the IEEE Signal Processing journal, in 2014. She has co-edited a book on 5G.