IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 6, 2018, accepted December 19, 2018, date of publication December 28, 2018,
date of current version January 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2890111

A Hybrid Model Based on Constraint OSELM, Adaptive Weighted SRC and KNN for Large-Scale Indoor Localization

HENGYI GAN[1], MOHD HARIS BIN MD KHIR[1], (Member, IEEE), GUNAWAN WITJAKSONO BIN DJASWADI[1], AND NORDIN RAMLI[2], (Senior Member, IEEE)

[1]Department of Electrical and Electronic Engineering, Universiti Teknologi PETRONAS, Seri Iskandar 32610, Malaysia
[2]Wireless Network and Protocol Research Laboratory, MIMOS Berhad, Kuala Lumpur 57000, Malaysia

Corresponding author: Hengyi Gan (heng_17005554@utp.edu.my)

This work was supported by the Universiti Teknologi PETRONAS under Grant 0153AA-H23.

ABSTRACT In this paper, a novel hybrid model based on the constraint online sequential extreme learning machine (COSELM) classifier with adaptive weighted sparse representation classification (WSRC) and k nearest neighbor (KNN) is proposed for the WiFi-based indoor positioning system. It is referred to as A Fast-Accurate-Reliable Localization System (AFARLS). AFARLS exploits the speed advantage of COSELM to reduce the computational cost, and the accuracy advantage of WSRC to enhance the classification performance, by utilizing KNN as the adaptive sub-dictionary selection strategy. The understanding is that the original extreme learning machine (ELM) is less robust against noise, while sparse representation classification (SRC) and KNN suffer a high computational burden when using the over-complete dictionary. AFARLS unifies their complementary strengths to resolve each other's limitation. In large-scale multi-building and multi-floor environments, AFARLS estimates a location that considers the building, floor, and position (longitude and latitude) in a hierarchical and sequential approach according to a discriminative criterion to the COSELM output. If the classifier result is unreliable, AFARLS uses KNN to achieve the best relevant sub-dictionary. The sub-dictionary is fed to WSRC to re-estimate the building and the floor, while the position is predicted by the ELM regressor. AFARLS has been verified on two publicly available datasets, the *EU Zenodo* and the *UJIIndoorLoc*. The experimental results demonstrate that AFARLS outperforms the state-of-the-art algorithms on the former dataset, and it provides near state-of-the-art performance on the latter dataset. When the size of the dataset increases remarkably, AFARLS shows that it can maintain its real-time high-accuracy performance.

INDEX TERMS Constraint online sequential extreme learning machine, *k* nearest neighbor, weighted sparse representation classification, WiFi-based IPS.

I. INTRODUCTION

The implementation of the Internet of Things (IoT) is in a wide diversity of fields ranging from smart infrastructure, healthcare applications, industrial automation to real-time monitoring and tracking, etc. The IoT-based object localization and tracking are considered as one of the recent active and immense developments of IoT applications [1]. Nonetheless, the localization and tracking system is not new. Since the first satellite navigation system was studied by the U.S. Navy using five satellites in 1960, massive developments towards the system have continued even as it reached the full operational status in 1995 [2]. Currently, the outdoor geolocation of an object is obtained from the Global Positioning System (GPS) which uses four or more satellites. Undeniably, the mature technology of the outdoor positioning system that relies on the GPS has had a tremendous impact on users' everyday lives. Examples include navigation, tracking, mapping and so forth. However, the GPS does not work well in indoor environments because it requires line-of-sight (LoS) measurement [3]. Hence, the precision of around 50 meters achieved by the GPS for the non-line-of-sight (NLoS) of reference objects in a complex indoor environment is very limited to commercial applications [4]. To resolve these limitations, various signals include WiFi, Bluetooth, RFID,

ultrasound, light and magnetic field have been investigated for IoT-based indoor positioning system (IPS).

The WiFi-based positioning technology is attracting present-day scientific and enterprise interests owing to its pervasive penetration of radio frequency signals and extensive deployment of WiFi enabled devices [3], [5]. According to [6], there are two types of WiFi based positioning technologies. They are the time and space attributes of received signal (TSARS)-based positioning technology and the energy attribute of received-signal strength (RSS)-based positioning technology. TSARS-based positioning technology can be based on the techniques such as Time of Arrival (ToA) [7], Angle of Arrival (AoA) [8] and Time Difference of Arrival (TDoA) [9] of the WiFi signal to determine the users' locations. AoA requires directional antennas or antenna arrays to extract the information of the angle of arrival signals. Meanwhile, ToA requires all the clocks of the target and the anchor nodes to be in precise synchronization. Unlike ToA, TDoA requires only the clock synchronization for the anchor nodes [4]. Since RSS of the WiFi signal is subjected to signal degradation as it traverses air over distances, the RSS-based positioning technology can adopt a range-based technique or a fingerprint-matching technique to locate the users. Both techniques require no additional positioning devices or clock synchronization. The range-based technique that is based on the trilateration method depends highly on the indoor radio propagation model [10], [11] and requires priori precise location information of the APs (anchor nodes). Due to it being very tough to well-establish a sophisticated indoor radio propagation model in a dynamic indoor environment, it suffers from low positioning accuracy. This difficulty arises due to the radio signal propagation being mainly affected by the interferences from the attenuation of the signal, multipath fading and shadowing effects etc. [6], [12]. On the contrary, the fingerprint-matching technique [13] requires no priori location information of any of the APs. It comprises the offline phase and the online phase [3], [6]. During the offline phase, the intensity of the signal strengths of different APs are collected with the MAC addresses at every location of Reference Point (RP) to establish a radio map. In the online phase, the real-time RSS obtained from the target node is compared with the radio map via specific fingerprint-based localization algorithms to estimate the most relevant position of the target node. Although the fingerprint-matching technique achieves higher accuracy [14], it requires tremendous setup and maintenance times, and different survey reduction algorithms [3], [15].

The k nearest neighbor (KNN) algorithm is one of the simplest and most effective among the fingerprint-based localization algorithms in supervised machine learning. It works by comparing the features of the testing data points with all the labeled samples from a training dataset. According to some prespecified distance metrics, the k nearest neighbors to the points are extracted from the training dataset and contribute equally for the final decision to label the points. In [16], the best among 51 distance metrics have been investigated with the KNN algorithm for the WiFi-based IPS. As a result, the indoor positioning based on the *Sorensen* distance with *powed* representation has been demonstrated to outperform the traditional KNN methods, like the 1-NN based on raw data and the most popular *Euclidean* distance. The main advantage of this algorithm is that the training phase is very minimal, apart from being simple and effective. Despite all of these benefits, the time of computational complexity is quite huge in the testing phase because of the necessity to determine the distance of each query instance to all of the training samples [17]. Further, it relies highly on the number of the training samples. In other words, it performs better with more reliable training samples, but it costs more time in the operational computation and requires larger memory to store the reference database. The other important factors that affect the performance are the selection of the appropriate k value and the decision rule in smoothing the k nearest neighbors.

Despite the WiFi-based IPS being able to establish high positioning accuracy through fingerprint-based localization algorithms like support vector machine (SVM) and KNN, these traditional algorithms, which are batch learning methods, pose some disadvantages like huge labor-cost calibration, heavy time consumption in either the training or the testing phases, and the recalibration required for different environments [3]. Recently, extreme learning machine (ELM) [18], [19] emerges as a very popular solution for large-scale applications due to its extremely fast learning speed. More advanced ELM algorithms are developed after that and have been applied for better localization performance. For instance, semi-supervised ELM (SELM) was proposed in [20] to include graph Laplacian regularization so that it did not depend too much on the labeled calibration data for a location estimation. In [21], fusion semi-supervised extreme learning machine (FSELM) was also proposed as a better semi-supervised learning approach to reduce the human calibration effort for indoor localization by considering the fusion information from WiFi and Bluetooth Low Energy (BLE) signals. Moreover, online sequential extreme learning machine (OSELM) [22] was developed and applied for the indoor localization in [23] to address the problems accordingly by using the traditional batch learning methods. The fast learning speed of OSELM has proven that it can help to reduce the intensive labor-cost and time-consuming site survey during the offline phase. Besides that, its online sequential learning ability can make the system to be more invulnerable to environmental dynamics. For the sake of the improved stability and generalization ability, the original ELM was optimized with the L2 regularization parameter [24]. For the sake of clarity, we refer to the ELM that uses L2 regularization as ELM-C in this paper. For better and lifelong localization service, the L2 regularization parameter was also introduced in OSELM, and it was referred to as COSELM [25]. COSELM was developed to overcome the fluctuation of the WiFi signal in the highly dynamic indoor environments due to the changing status of the door [26], the changing status of the relative humidity, and the people's

presence [27]. Therefore, it can maintain a viable system in a longtime running performance. Although the COSELM classifier performs extremely fast, the original ELM itself handles noise poorly. To make the classifier more robust against noise, sparse representation classification (SRC) is of particular interest to [28] by combining the ELM-C with the adaptive SRC for image classification, referred to as EA-SRC. The EA-SRC model is designed to inherit the respective excellent characteristics of ELM which has low computational cost, and SRC which has high prediction accuracy.

For long-lasting stable large-scale indoor localization in a multi-floor single building environment, we have proposed a novel indoor localization hybrid model, AFARLS, which combines ELM, SRC and KNN. AFARLS estimates the location that consists of floor and position (longitude and latitude) in a hierarchical and sequential approach. Instead of using the original ELM as a classifier, AFARLS employs the COSELM classifier that utilizes the leave-one-out (LOO) cross-validation scheme for the optimal regularization parameter selection from [28]. Similar to the concept of EA-SRC, AFARLS will only involve SRC if the classification results in identifying the floor level are unreliable according to a discriminative criterion to the COSELM output. As mentioned previously, using an over-complete dictionary for SRC poses a high computational complexity and lack of adaptability. Thus, an adaptive sparse domain selection strategy is very encouraged to resolve the negative effects of uncorrelated classes of fingerprints. In AFARLS, we employ a very different adaptive sparse domain selection strategy for each query RSS fingerprint. We integrate AFARLS with KNN based on the *Sorensen* distance metric with *powed* data representation to achieve the best relevant sub-dictionary. The sub-dictionary is subsequently fed to SRC to classify the unreliable results again. Since the performance of the KNN-based classification can be improved by introducing a weighting scheme for the nearest neighbors, we develop WSRC working with KNN to strengthen the classification results as it is insufficient to distinguish the RSS fingerprint through the residual alone. The noteworthy feature of WSRC is based on the conceptual basics of weighted k nearest neighbor (WKNN) that emphasizes close neighbors more heavily. Rather than being based on the distances to the query, WSRC is based on the residual to the query. Meanwhile, AFARLS utilizes the same sub-dictionary generated from KNN to train ELM-C, and later to perform multi-target regression to estimate the position. Dealing with the multi-building and multi-floor indoor environments, the real-world coordinates consist of the building, floor and position (longitude and latitude) as a location. Accordingly, an additional label of the building identification is considered in COSELM classifier for multi-label classification. We verity the proposed model in large-scale indoor environments with two different databases, the *EU Zenodo* dataset [29] which is a five-floor building with almost $22570m^2$ total surface area, and the *UJIndoorLoc* dataset [30] which covers a surface of almost $108703m^2$ including three buildings with either four or five floors. Experimental results exhibit the real-time high-accuracy localization performance of AFARLS in a large-scale multi-building and multi-floor environment, together with its long-term feasibility by leveraging online incremental measurements to continuously update the model. In short, the main contributions of this paper are as follows:

- We design a novel state-of-the-art localization algorithm with an online sequential learning ability, combining COSELM that based on the LOO cross-validation scheme and WSRC to complement each other in computational complexity and classification accuracy.
- We propose COSELM as a novel clustering-based approach in combination with KNN based on the *Sorensen* distance metric with *powed* data representation as an adaptive sparse domain selection strategy to achieve the best relevant sub-dictionary.
- We develop WSRC to emphasize close neighbors from the sub-dictionary more heavily according to the residual to the query, rather than the distances to the query to strengthen the classification performance.

The rest of this work is organized as follows. Section II reviews the studies relevant to our work. Section III describes and analyzes the system architecture. Section IV evaluates the performance of the system model under large-scale indoor localization environments. Finally, Section V concludes the paper.

II. BACKGOUND INFORMATION

In this section, we review the studies pertinent to the framework of our proposed system architecture, namely AFARLS. The scope of the studies focuses on KNN, ELM and SRC algorithms, together with their respective enhancements in order to facilitate the understanding of our analytic model.

A. KNN

The KNN algorithm is widely utilized for classification, though it can be used for regression. It extracts the k training samples that are most identical or nearest to a test sample. After that, the test sample is labeled based on the majority label among the k nearest neighbors. Given a training database, $(\boldsymbol{x}_j, \boldsymbol{t}_j) \in \boldsymbol{R}^n \times \boldsymbol{R}^m$ used in the supervised machine learning, $\boldsymbol{x}_j$ is the j-th $n \times 1$ training input vector (feature), $\boldsymbol{t}_j$ is the j-th $m \times 1$ training target vector (label) and $j = 1, 2, \ldots, N$ samples. The output is to determine the k nearest neighbors $\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_k\}$ in $\{\boldsymbol{x}_j\}_{j=1}^{N}$ to an unlabeled test sample $\boldsymbol{y} \in \boldsymbol{R}^{n \times 1}$, and assign $\boldsymbol{y}$ based on the corresponding labels $\{\boldsymbol{t}_{\boldsymbol{r}_1}, \ldots, \boldsymbol{t}_{\boldsymbol{r}_k}\}$. In short, KNN classifies the pattern $\boldsymbol{y}$ to one of the possible M classes of the problem at hand [31] as shown in (1).

$$\boldsymbol{y} \leftarrow \arg \max_{c \in \{1,2,\ldots,M\}} [\sum_{l=1}^{k} I_c(\boldsymbol{r}_l)] \tag{1}$$

where $\boldsymbol{r}_l \in \boldsymbol{R}^{n \times 1}$ is the l^{th} neighbor in $\boldsymbol{x}$ nearest to $\boldsymbol{y}$ according to $distance_{metric}(\boldsymbol{x}, \boldsymbol{y})$, $l = 1, 2, \ldots, k$, $c = 1, 2, \ldots, M$, and $I_c(\boldsymbol{r}_l)$ is the indicator function of the l^{th} nearest neighbor

Algorithm 1 KNN-Based Classification

Input: A training data set $\{(\boldsymbol{x}_j, \boldsymbol{t}_j)\}_{j=1}^{N}$, a test sample $\boldsymbol{y}$, k parameter, and $distance_{metric}(\cdot)$.
Output: Class label of $\boldsymbol{y}$
1: Compute the similarity rates between $\boldsymbol{x}$ and $\boldsymbol{y}$ according to a distance metric
2: Select k nearest neighbors that have the highest similarity rates
3: Label($\boldsymbol{y}$) = $\arg \max_{c\in\{1,2,\ldots,M\}} [\sum_{l=1}^{k} I_c(\boldsymbol{r}_l)]$

belonging to class c.

$$I_c(\boldsymbol{r}_l) = \begin{cases} 1 & \text{if } \boldsymbol{r}_l \text{ belongs to class } c \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

By introducing different weights to every nearest neighbor, KNN is developed to WKNN. The weighting scale depends on the distance between the test sample and the nearest neighbor. The primary effect of the weighting scheme is that its decision rule on the k nearest neighbors has less significance on the majority vote than the closer neighbors. As a result, the pattern $\boldsymbol{y}$ is assigned to one of the possible M classes in which the rule is not merely based on the closest distance to the query, but also the most frequent among its k nearest neighbors, as shown in (3).

$$\boldsymbol{y} \leftarrow \arg \max_{c\in\{1,2,\ldots,M\}} [\sum_{l=1}^{k} I_c(\boldsymbol{r}_l) W_{\boldsymbol{r}_l}] \tag{3}$$

$$W_{\boldsymbol{r}_l} = \frac{1}{distance_{metric}(\boldsymbol{y}, \boldsymbol{r}_l)} \tag{4}$$

Depending on the types of the distance metrics, the performance of KNN is affected. Among the distance metrics studied in [16], (5) is the most popular and it is also referred to as the traditional KNN approach, while (6) has been recently proven to be the most efficient for localization.

$$distance_{euclidean}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[2]{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{5}$$

$$distance_{sorensen}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{i=1}^{n} |x_i - y_i|}{\sum_{i=1}^{n} (x_i - y_i)} \tag{6}$$

$$distance_{neyman}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i} \tag{7}$$

B. ELM

ELM is proposed based on a single-hidden layer feedforward neural network (SLFN) architecture [18], and the latter extended to the generalized feedforward network. ELM itself is a supervised batch learning method. Its notable merit lies in it randomly selects the hidden node parameters (input weights and bias), and then it determines only the output weights. For N arbitrary distinct data samples, $\{(\boldsymbol{x}_j, \boldsymbol{t}_j)\}_{j=1}^{N}$ is used in the supervised batch learning, where $\boldsymbol{x}_j = [\boldsymbol{x}_{j1}, \boldsymbol{x}_{j2}, \ldots, \boldsymbol{x}_{jn}]$ is a training input vector, n is the dimension of the feature, $\boldsymbol{t}_j = [\boldsymbol{t}_{j1}, \boldsymbol{t}_{j2}, \ldots, \boldsymbol{t}_{jm}]$ is a training target vector, and m is the dimension of the label. If a SLFN with additive L hidden nodes can approach these N samples with no error, the output function of the network is

$$\begin{aligned} f_L(\boldsymbol{x}_j) &= \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j) = \boldsymbol{s}_j \\ \boldsymbol{w}_i &\in \boldsymbol{R}^n, \quad b_i \in R, \ \boldsymbol{x}_j \in \boldsymbol{R}^n, \ \boldsymbol{\beta}_i \in \boldsymbol{R}^m, \ \boldsymbol{s}_j \in \boldsymbol{R}^m \\ & i = 1, 2, \ldots, L; \quad j = 1, 2, \ldots, N \end{aligned} \tag{8}$$

where $\boldsymbol{w}_i = [\boldsymbol{w}_{i1}, \boldsymbol{w}_{i2}, \ldots, \boldsymbol{w}_{in}]$ is the input weight vector connecting the i^{th} hidden node to the input nodes and b_i is the bias of the i^{th} hidden node, $\boldsymbol{\beta}_i = [\boldsymbol{\beta}_{i1}, \boldsymbol{\beta}_{i2}, \ldots, \boldsymbol{\beta}_{im}]^T$ is the output weight vector connecting the i^{th} hidden node to the output nodes, $\boldsymbol{s}_j = [\boldsymbol{s}_{j1}, \boldsymbol{s}_{j2}, \ldots, \boldsymbol{s}_{jm}]$ is the actual network output vector with respect to $\boldsymbol{x}_j$, and $G(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j)$ is the activation function. Since the hidden nodes are additive,

$$G(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j) = g(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) \tag{9}$$

Equation (8) can be summarized as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{10}$$

where,

$$\begin{aligned} \mathbf{H} &= \begin{bmatrix} g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_1 + b_1) & \cdots & g(\boldsymbol{w}_L \cdot \boldsymbol{x}_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_N + b_1) & \cdots & g(\boldsymbol{w}_L \cdot \boldsymbol{x}_N + b_L) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{h}(\boldsymbol{x}_N) \end{bmatrix}_{N\times L} \\ \boldsymbol{\beta} &= \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L\times m}, \quad \mathbf{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{N\times m} \end{aligned} \tag{11}$$

Here $\mathbf{H}$ is called the hidden layer output vector. The notable merit of the batch ELM algorithm is the random generation of the hidden parameters of $\boldsymbol{w}_i$ and b_i without tuning during training. Therefore, (10) becomes a linear system, and the $\boldsymbol{\beta}$ is obtained by solving the following least squares problem to minimize the error between $\boldsymbol{t}_j$ and $\boldsymbol{s}_j$.

$$\min_{\boldsymbol{\beta}} ||\mathbf{H}\boldsymbol{\beta} - \mathbf{T}|| \tag{12}$$

Here, the $\boldsymbol{\beta}$ is estimated by

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T} \tag{13}$$

where $\mathbf{H}^{\dagger}$ is the Moore-generalized inverse or the pseudo-inverse of $\mathbf{H}$, and it can be calculated as [18], i.e.

$$\mathbf{H}^{\dagger} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T \tag{14}$$

However, the pseudo-inverse suffers from numerical instability. Hence, ELM is enhanced to ELM-C by using the regularized least squares from [32] to guarantee minimum

Algorithm 2 ELM-C Based Classification

Input: A training data set $\{(\boldsymbol{x}_j, \boldsymbol{t}_j)\}_{j=1}^{N}$, a test sample $\boldsymbol{y}$, activation function$G(\cdot)$, and L hidden nodes.
Output: Class label of $\boldsymbol{y}$
1: Randomly generate parameters $\{\boldsymbol{w}_i, b_i\}$
2: Calculate $\mathbf{H}$
3: Compute λ_{opt}
4: Determine $\hat{\boldsymbol{\beta}}$
5: Compute the actual network output, $\boldsymbol{s}$ w.r.t $\boldsymbol{y}$
6: Label($\boldsymbol{y}$) = arg $\max_{c\in\{1,2,\ldots,M\}}$ ($\boldsymbol{s}$)

error. In ELM-C, the $\boldsymbol{\beta}$ is estimated by either (15) or (16). When the training data sets are very large $N \geq L$,

$$\hat{\boldsymbol{\beta}} = \left(\lambda_{opt}\mathbf{I} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \tag{15}$$

else,

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^T\left(\lambda_{opt}\mathbf{I} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \tag{16}$$

where $\mathbf{I}$ is the identity matrix and λ_{opt} is the optimal regularization parameter that can be determined from the range of $[-\infty, \infty]$. Since different regularization parameter λ generates different generalization performance, the LOO cross-validation approach in [28] is adopted as an effective method for the parameter optimization for the search of λ_{opt}, by considering a tradeoff between the training error term and λ. Unless stated otherwise, $\lambda \in [e^{-4}, e^{4}]$ is selected as the candidate set of the regularization parameters in this paper to determine λ_{opt}, and then the corresponding optimal $\hat{\boldsymbol{\beta}}$. This optimization is beyond the paper's scope.

The online learning version of ELM is OSELM. It consists of two phases, an initialization phase and a sequential learning phase. The sequential learning phase updates the out-of-date model with online incremental data that may come in chunk-by-chunk or one-by-one. Given the initial large chunk of data set, $X_0 = \{(\boldsymbol{x}_j, \boldsymbol{t}_j)\}_{j=1}^{N_0}$, where N_0 is the number of the initial training data in this chunk, the initial estimated $\hat{\boldsymbol{\beta}}^0$ is

$$\hat{\boldsymbol{\beta}}^0 = \mathbf{K}_0^{-1}\mathbf{H}_0^T\mathbf{T}_0 \tag{17}$$

where,

$$\mathbf{K}_0 = \mathbf{H}_0^T\mathbf{H}_0 \tag{18}$$

When the new chunk of data set $X_1 = \{(\boldsymbol{x}_j, \boldsymbol{t}_j)\}_{j=N_0+1}^{N_0+N_1}$ arrives, N_1 is the number of new training data, the estimated $\hat{\boldsymbol{\beta}}^1$ is

$$\hat{\boldsymbol{\beta}}^1 = \hat{\boldsymbol{\beta}}^0 + \mathbf{K}_1^{-1}\mathbf{H}_1^T(\mathbf{T}_1 - \mathbf{H}_1\hat{\boldsymbol{\beta}}^0) \tag{19}$$

where,

$$\mathbf{K}_1 = \mathbf{K}_0 + \mathbf{H}_1^T\mathbf{H}_1, \quad \mathbf{K}_0 = \mathbf{H}_0^T\mathbf{H}_0 \tag{20}$$

Consider p as the parameter that denotes the number of updating times of the chunks of data presented for the online sequential learning. In general, the solution of OSELM after $p+1$ times of incremental learning is

$$\hat{\boldsymbol{\beta}}^{(p+1)} = \hat{\boldsymbol{\beta}}^{(p)} + \mathbf{K}_{p+1}^{-1}\mathbf{H}_{p+1}^T(\mathbf{T}_{p+1} - \mathbf{H}_{p+1}\hat{\boldsymbol{\beta}}^{(p)}) \tag{21}$$

where,

$$\mathbf{K}_{p+1} = \mathbf{K}_p + \mathbf{H}_{p+1}^T\mathbf{H}_{p+1}, \quad \mathbf{K}_0 = \mathbf{H}_0^T\mathbf{H}_0 \tag{22}$$

The stability and generalization performance of OSELM can be improved in a similar way as performed for ELM by utilizing the regularized least squares to replace the pseudo-inverse. We refer it to as COSELM. In COSELM, given the initial data $X_0 = \{(\boldsymbol{x}_j, \boldsymbol{t}_j)\}_{j=1}^{N_0}$, the initial estimated $\hat{\boldsymbol{\beta}}^0$ becomes

$$\hat{\boldsymbol{\beta}}^0 = \left(\lambda_{opt}\mathbf{I} + \mathbf{H}_0^T\mathbf{H}_0\right)^{-1}\mathbf{H}_0^T\mathbf{T}_0 = \mathbf{K}_0^{-1}\mathbf{H}_0^T\mathbf{T}_0 \tag{23}$$

where,

$$\mathbf{K}_0 = \lambda_{opt}\mathbf{I} + \mathbf{H}_0^T\mathbf{H}_0 \tag{24}$$

and the estimated output weight of COSELM after $p+1$ times incremental learning is

$$\hat{\boldsymbol{\beta}}^{(p+1)} = \hat{\boldsymbol{\beta}}^{(p)} + \mathbf{K}_{p+1}^{-1}\mathbf{H}_{p+1}^T(\mathbf{T}_{p+1} - \mathbf{H}_{p+1}\hat{\boldsymbol{\beta}}^{(p)}) \tag{25}$$

where,

$$\mathbf{K}_{p+1} = \mathbf{K}_p + \mathbf{H}_{p+1}^T\mathbf{H}_{p+1}, \quad \mathbf{K}_0 = \lambda_{opt}\mathbf{I} + \mathbf{H}_0^T\mathbf{H}_0 \tag{26}$$

Compared to OSELM, COSELM becomes OSELM when λ_{opt} approximates 0, making the $\lambda_{opt}\mathbf{I} + \mathbf{H}_0^T\mathbf{H}_0 \approx \mathbf{H}_0^T\mathbf{H}_0$. Hence, COSELM is a special case of OSELM, whereas OSELM is a sepcial case of ELM. The extension of ELM to COSELM trumps the traditional supervised batch learning ELM, because it achieves better generalization performance and is adaptive to changes.

C. SRC

Given a database with N arbitrary distinct data samples, $\{\boldsymbol{X}_c\}_{c=1}^{M}$ for M classes, $\boldsymbol{X}_c \in \boldsymbol{R}^{n\times k_c}$ that has n-dimensional features and k_c training input samples belonging to the class c, the complete dictionary is formed as $\boldsymbol{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M]$. In fact, the high dimensional and correlated training features of $\boldsymbol{X}$ can be sparsely represented in sparse dictionary $\boldsymbol{D}$ that contains a feature vector of much lower density. In general, a sparse representation of $\boldsymbol{X}$ can be formalized as follows.

$$\boldsymbol{X} = \boldsymbol{D}\boldsymbol{v} + \boldsymbol{r} \tag{27}$$

where $\boldsymbol{v}$ represents the sparse coefficient vector and $\boldsymbol{r}$ denotes the residual after sparse representation. To label a test sample of $\boldsymbol{X}_t$, the following two optimization problems can be used to solve for the optimal $\boldsymbol{v}$.

$$\hat{\boldsymbol{v}} = \arg\min_{\boldsymbol{v}} \|\boldsymbol{v}\|_0 \quad s.t.\ \boldsymbol{D}\boldsymbol{v} = \boldsymbol{X}_t \tag{28}$$

$$\hat{\boldsymbol{v}} = \arg\min_{\boldsymbol{v}} \|\boldsymbol{v}\|_1 \quad s.t.\ \|\boldsymbol{D}\boldsymbol{v} - \boldsymbol{X}_t\|_2^2 < \varepsilon \tag{29}$$

where $\|\cdot\|_0$ is l_0-norm, $\|\cdot\|_1$ is l_1-norm and ε is a small error tolerance in $\boldsymbol{X}_t$. The latter is preferable for the minimization

Algorithm 3 COSELM-Based Classification

Input: A training data set $\{(\boldsymbol{x}_j, \boldsymbol{t}_j)\}_{j=1}^{N}$, a test sample $\boldsymbol{y}$, activation function $G(\cdot)$, L hidden nodes, initial chunk of data size N_0, subsequent chunks of online data size N_{p+1}.
Output: Class label of $\boldsymbol{y}$
Initialization Phase
1: Randomly generate parameters $\{\boldsymbol{w}_i, b_i\}$
2: Calculate $\mathbf{H}_0$
3: Compute λ_{opt}
4: Determine $\hat{\boldsymbol{\beta}}^{0}$
5: Set $p = 0$
Online Sequential Phase
6: **If** the $(p+1)^{th}$ chunk of data arrives, **then**
  a: Calculate $\mathbf{H}_{p+1}$
  b: Update $\hat{\boldsymbol{\beta}}^{(p+1)}$
  c: Set $p = p + 1$
  d: Go to Step 6
7: **else**
  a: Go to Step 9
8: **end if**
9: Compute the actual network output, $\boldsymbol{s}$ w.r.t $\boldsymbol{y}$
10: Label($\boldsymbol{y}$) = arg $\max_{c\in\{1,2,\ldots,M\}}$ ($\boldsymbol{s}$)

Algorithm 4 SRC-Based Classification

Input: A dictionary with M classes $\boldsymbol{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_M]$, a test sample $\boldsymbol{X}_t$.
Output: Class label of $\boldsymbol{X}_t$
1: Normalize the column vectors of $\boldsymbol{X}$ in unit of l_2-norm
2: Solve the optimization problem (29)
3: Calculate the residuals, $r_c(\boldsymbol{X}_t) = \left\|\boldsymbol{D}\delta_c(\hat{\boldsymbol{v}}) - \boldsymbol{X}_t\right\|_2^2$, where $c = 1, 2, \ldots, M$
4: Label $(\boldsymbol{X}_t)$ = arg $\min_{c\in\{1,2,\ldots,M\}}$ $r_c(\boldsymbol{X}_t)$

problem in SRC due to the NP-hard problem of the former [28]. With the optimal $\boldsymbol{v}$, the characteristic function $\delta_i(\hat{\boldsymbol{v}})$ in SRC generates a new vector that only possesses nonzero coefficients in $\hat{\boldsymbol{v}}$ associated to the respective c-th class. If it belongs to the class c, the residual of the class c, $r_c(\boldsymbol{X}_t) = \left\|\boldsymbol{D}\delta_c(\hat{\boldsymbol{v}}) - \boldsymbol{X}_t\right\|_2^2$ is the minimum after it is approximately represented by the sparse dictionary of that class; otherwise the corresponding residual would be relatively huge.

$$\begin{aligned} &\text{Label } (\boldsymbol{X}_t) = \arg \min_{c\in\{1,2,\ldots,M\}} r_c(\boldsymbol{X}_t) \\ &\text{with } r_c(\boldsymbol{X}_t) = \left\|\boldsymbol{D}\delta_c(\hat{\boldsymbol{v}}) - \boldsymbol{X}_t\right\|_2^2 \end{aligned} \tag{30}$$

III. SYSTEM ARCHITECTURE

In this section, we present the architecture of the proposed localization system model. AFARLS has three phases: offline training phase, online sequential learning phase and online localization phase or online testing phase. Three dimensional real-world coordinates in which floor and position (longitude and latitude) are considered as a location in a multi-floor building. The fingerprint database $D = \{(\boldsymbol{x}_j, \boldsymbol{t}_j)\}_{j=1}^{N}$ contains N arbitrary data samples of WiFi fingerprints. The fingerprint database is established through the popular collaborative or crowdsourced method and collected using different mobile devices. More precisely, the training input vector, $\boldsymbol{x}_j = [\boldsymbol{x}_{j1}, \boldsymbol{x}_{j2}, \ldots, \boldsymbol{x}_{jn}]$ contains the fingerprints and the training target vector, $\boldsymbol{t}_j = [\boldsymbol{t}_{j1}, \boldsymbol{t}_{j2}, \ldots, \boldsymbol{t}_{jm}]$ contains the physical reference locations, where n is the number of APs present in a building and m is the number of dimensions for a defined location. Since the test bed is a single building, we set $m = 3$ for multi-label classification of the floor, the longitude and the latitude. Before the offline training phase, the reference locations are rounded to the nearest integers. The non-heard AP fingerprints are represented with 0. The fingerprints in which the RSS values are less than the RSS threshold, τ, are also represented with 0. Next, the lowest RSS value, RSS_{min}, is identified by considering all fingerprints and the APs of the database. New data representation is performed according to (31) based on [16]. As a result, the lowest value is 0 for the non-heard or very weak-signal AP fingerprints, and the higher values are for the stronger signals.

$$\begin{aligned} Positive_i\left(\boldsymbol{x}_j\right) &= RSS_i - (RSS_{min} - 1) \\ i &= 1, 2, \ldots, n; \quad j = 1, 2, \ldots, N \end{aligned} \tag{31}$$

During the offline training phase, the COSELM classifier in AFARLS trains the fingerprint database at every reference location. After the offline training phase, we can still update the existing model during the online sequential learning phase by leveraging the online incremental data. In the online localization phase, given a test sample $\boldsymbol{f} \in \boldsymbol{R}^n$, COSELM classifies $\boldsymbol{f}$ to the location labels which are the estimated coordinates that consist of the information of the floor level and the position (longitude, latitude). These labels are classified based on the three highest confident scores in a three-column matrix of the actual network output vector, $\boldsymbol{s} \in \boldsymbol{R}^m$. More specifically, in a single building indoor environment, the first column vector $\boldsymbol{s}_1$ and the second column vector $\boldsymbol{s}_2$ contain the confident scores of the longitude and the latitude respectively, while the last column vector $\boldsymbol{s}_3$ contains the confident scores of the floor. If it is to predict the floor level associated to $\boldsymbol{f}$, the floor is classified according to the highest confident score in $\boldsymbol{s}_3$.

Since ELM itself is known to be less robust against noise in classification tasks, the SRC-based classification is decided based on a discriminative criterion of whether to be included as an additional step to predict the floor level after COSELM. The criterion measures the difference between the first and the second highest confident scores in $\boldsymbol{s}_3$. We denote the difference as $\delta_3 = \boldsymbol{s}_3^F - \boldsymbol{s}_3^S$, where $\boldsymbol{s}_3^F$ and $\boldsymbol{s}_3^S$ are the first and the second highest confident scores in $\boldsymbol{s}_3$ respectively. If δ_3 falls below an appropriate chosen positive threshold, σ, the criterion is then satisfied and the misclassification rate of COSELM under this condition tends to be higher. Therefore, the floor level will not be estimated according to the highest

confident score in $\boldsymbol{s}_3$. This is because of the fingerprint with a small δ_3 is contaminated with a larger noise signal that can adversely affect the result more than the one with a big δ_3. For these noisy test fingerprints $\boldsymbol{f}$, SRC is incorporated following after COSELM.

Generating an adaptive sub-dictionary associated to $\boldsymbol{f}$ is an important step for SRC to reduce the overall computation burden. If the discriminative criterion is satisfied, the adaptive sub-dictionary is used by SRC to predict the floor and is fed to the ELM-C regressor to estimate the position. Otherwise, it is used by ELM-C to predict the position. First, we alter the usage of COSELM from classifying $\boldsymbol{f}$ to the location labels to where the user belongs, to clustering a group of relatively closely associated training fingerprints with the corresponding reference locations. Instead of having one set of the location labels as the final output, COSELM clusters q pairs of position labels corresponding to the respective top q highest confident scores in $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. In order to enhance the probability of the classification rate, we increase the size of the cluster group by establishing a new larger cluster of the coordinates from the database whereby either their longitude or latitude labels match to the q pairs of the position labels. If the discriminative criterion is satisfied, the relevant fingerprints, $\boldsymbol{p}$ associated to these coordinates are extracted from the database. Otherwise, we further reduce the size of the cluster by extracting only the relevant fingerprints $\boldsymbol{p}$, in which the corresponding coordinates have the floor that matches the predicted floor determined from COSELM. Although the size of the newly constructed subset has been greatly reduced compared to the original database, the direct application of the SRC-based classification to predict the floor level and the direct application of ELM-C based regression to estimate the position are time-consuming. Hence, it is appropriate to construct a fixed size of k, which is the smaller subset of the extracted training samples. To address this issue, we utilize the KNN-based classification as the adaptive sub-dictionary selection strategy.

Despite employing KNN at the very beginning to determine the optimal k nearest neighbors from the database, we propose it after COSELM. This can be explained in terms of the reduced computational costs when KNN works with clustering algorithms such as the K-means clustering approach [33] and the support vector machine-based clustering approach [34]. Instead of using the initial large database, KNN figures out the nearest neighbors from a cluster which contains fewer samples which are more correlated to each other. Since COSELM has demonstrated promising features in computational speed and learning new data continuously, we adopt COSELM as the clustering algorithm for KNN to generate the adaptive sub-dictionary which contains the optimal k nearest fingerprints, $\boldsymbol{o}$ to $\boldsymbol{f}$. As pointed out previously, the recent developed KNN based on the *Sorensen* distance metric with *powed* data representation generates the best result compared to the traditional KNN. Thus, we adopt KNN that is based on this configuration. First, the highest RSS value, RSS_{max}, is identified by considering the fingerprints, $\boldsymbol{p}$ and $\boldsymbol{f}$. Then, we convert the RSS values of $\boldsymbol{p}$ according to (32) and return $\hat{\boldsymbol{p}}$.

$$Powed_i(\boldsymbol{p}) = \frac{(\boldsymbol{p})^{\alpha}}{(RSS_{max})^{\alpha}} = \hat{\boldsymbol{p}}, \quad i = 1, 2, \ldots, n \tag{32}$$

where α is a mathematical constant, set to e. With $\hat{\boldsymbol{p}}$, KNN based on the *Sorensen* distance metric is performed to generate the adaptive sub-dictionary that contains $\boldsymbol{o}$ in which the RSS values are positive. For the sake of clarity, we refer to these steps, starting from the output of COSELM to KNN, as q-by-k clustering. After q-by-k clustering, we can perform the SRC-based classification to identify the floor by using the adaptive sub-dictionary that contains the fingerprints, $\boldsymbol{o}$, provided that the discriminative criterion is satisfied. Otherwise, the floor is directly determined from COSELM. On the other hand, the position is estimated from ELM-C based on the fingerprints in which the corresponding coordinates belong to the predicted floor from the adaptive sub-dictionary.

SRC sparsely represents parts of the nonzero elementary features of the fingerprints to an adaptive sparse dictionary that has much lower density for every floor, because of the high-dimensional feature of the fingerprints is compressible. For the floor level associated to $\boldsymbol{f}$ classified as the class of c, $\boldsymbol{f}$ should be approximately represented by the sparse dictionary of the c-th floor. In other words, for $\boldsymbol{f}$ that cannot be approximately represented by a floor's sparse dictionary, it is considered not belonging to this floor. Since the performance of KNN can be improved by introducing a weighting scheme for the nearest neighbors, we develop WSRC working with KNN based on the *Sorensen* distance metric with *powed* data representation to strengthen the classification results as it is insufficient to distinguish the RSS fingerprint through the residual alone. The noteworthy feature of WSRC is that it emphasizes close neighbors more heavily from the adaptive sub-dictionary, similar to the basic idea of WKNN. Rather than being based on the distances to the query, WSRC is based on the residual to the query. At first, the corresponding residual to each floor is converted to weightages. Based on the k neighbors, the frequency to each floor's vote is counted, so that the majority vote of the neighbors is considered as the metric for the classification accuracy. The primary effect of the weighting scheme on the WSRC is that its decision rule on the neighbors has less significance on the majority vote than the neighbors that have the smallest residuals. As a result, $\boldsymbol{f}$ is assigned to one of the possible floors in which the rule is not merely based on the smallest residual to the query, but also the most frequent among its k nearest neighbors. WSRC classifies $\boldsymbol{f}$ to one of the possible M classes of the floors of the problem at hand given as in (33).

$$\text{Label}(\boldsymbol{f}) = \arg \max_{c \in \{1,2,\ldots,M\}} [\sum_{l=1}^{k} I_c(\boldsymbol{o}_l) w_c(\boldsymbol{f})] \tag{33}$$

$$I_c(\boldsymbol{o}_l) = \begin{cases} 1 & \text{if } \boldsymbol{o}_l \text{ belongs to class } c \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

Algorithm 5 WSRC-Based Classification

Input: A dictionary with k samples $\boldsymbol{o} = [\boldsymbol{o}_1, \ldots, \boldsymbol{o}_k]$, a test sample $\boldsymbol{f}$.

Output: Class label of $\boldsymbol{f}$

1: Normalize the column vectors of $\boldsymbol{o}$ in unit of l_2-norm

2: Solve the optimization problem,
$\hat{\boldsymbol{v}} = \arg\min_{\boldsymbol{v}} \|\boldsymbol{v}\|_1 \quad s.t. \ \|\boldsymbol{D}\boldsymbol{v} - \boldsymbol{f}\|_2^2 < \varepsilon$

3: Calculate the residuals, $r_c(\boldsymbol{f}) = \left\|\boldsymbol{D}\delta_c(\hat{\boldsymbol{v}}) - \boldsymbol{f}\right\|_2^2$, where $c = 1, 2, \ldots, M$

4: Convert the residuals to weightages, $w_c(\boldsymbol{f})$

5: Label $(\boldsymbol{f}) = \arg \max_{c \in \{1,2,\ldots,M\}} [\sum_{l=1}^{k} I_c(\boldsymbol{o}_l) w_c(\boldsymbol{f})]$

$$w_c(\boldsymbol{f}) = \frac{1}{r_c(\boldsymbol{f})} \tag{35}$$

where $I_c(\boldsymbol{o}_l)$ is the indicator function of the l^{th} fingerprint in $\boldsymbol{o}$ corresponding to class c, w_c is the c-th weightage corresponding to the c-th residual, $c = 1, 2, \ldots, M$, and $l = 1, 2, \ldots, k$. In short, we consider a hybrid classifier model to predict the floor level in AFARLS which involves COSELM, KNN and WSRC. On the other hand, the position is determined from ELM-C after we have classified $\boldsymbol{f}$ to the floor by using the fingerprints in which the corresponding coordinates belong to the predicted floor from the adaptive sub-dictionary $\boldsymbol{o}$. Because of the sample size of $\boldsymbol{o}$ is relatively small compared to the original database, the results from the ELM-C and COSELM regressors are not much different. Hence, we utilize ELM-C to train $\boldsymbol{o}$, and compute the position associated to $\boldsymbol{f}$. L_{COSELM} and L_{ELM-C} are defined to denote the number of hidden nodes for the COSELM classifier and the ELM-C regressor respectively. Dealing with multi-building and multi-floor indoor environments, we consider a location has the real-world coordinates which consist of the building, the floor and the position (longitude and latitude). Accordingly, an additional label, which is the building identification, is required for multi-label classification in COSELM ($m = 4$). The subsequent classification steps that are customized to identify the building are very identical to the hybrid classifier model that we have discussed before to classify $\boldsymbol{f}$ to the floor. It is worth pointing out that it will involve KNN and WSRC if δ_4 from the output vector $\boldsymbol{s}_4$ does not satisfy the criterion. To distinguish between the thresholds, σ, belonging to the floor and the building, we use σ_{floor} and $\sigma_{building}$. In summary, the detailed pseudo-code and overview of AFARLS developed for a multi-building and multi-floor indoor environment are presented in algorithm 6 and Figure 1.

IV. EXPERIMENT RESULTS AND DISCUSSION

This section presents the experimental results and discussions. The results are obtained by running the experiments on a PC, which has an Intel Core i7-4700MQ CPU at 2.40GHz and 8GB RAM. Two large-scale publicly available Wi-Fi crowdsourced fingerprint datasets are used to verify the actual effect of the system model proposed in this paper in a multi-floor single building indoor environment and a realistic multi-floor multi-building indoor environment. The datasets are the *EU Zenodo* database and the *UJIIndoorLoc* database.

The *EU Zenodo* database was created in 2017 at Tampere University of Technology. It covers a five-floor building that contains 822 rooms in total and has about $22570m^2$ of footprint area of about $208m$ length and $108m$ width. The database consists of a total of 4648 Wi-Fi fingerprints and the corresponding reference locations. These locations contain the floor levels, the longitude and the latitude coordinates (in meters). From the database, it is split into the training set and the testing set. The former and the latter contains 697 and 3951 WiFi fingerprints respectively. Each row of the WiFi fingerprint is represented by a 992-column vector of MAC addresses which contains the default value of $+100dBm$ for those non-heard APs. If the MAC address is received, then it has a negative level of the RSS value (in dBm). All the measurements were reported from 21 user devices which could support both the 2.4GHz and 5GHz frequency range. On the other hand, the *UJIIndoorLoc* database was created in 2013 at Universitat Jaume I. It was reported from more than 20 different users and 25 Android devices. It covers three buildings with four or five floors. The total cover footprint area is almost $108703m^2$. The database contains 21048 WiFi fingerprints with the corresponding labels of the building, the floor and the real-world longitude and latitude coordinates (in meters) collected at pre-defined locations. Furthermore, the database is split into the training set and the validation set. The former contains 19937 training fingerprints and the latter contains 1111 validation fingerprints. Since the publicly available *UJIIndoorLoc* does not provide the testing samples which are made available only for the competitors at EvAAL [30], we use the validation data as the testing data. Each row of the WiFi fingerprints is represented by a 520-column vector of MAC addresses which contains the default value of $+100dBm$ for those non-heard APs. If the MAC address is received, then it has a negative level of the RSS value (in dBm).

The core of the performance evaluation considers the training time, testing time, training accuracy and testing accuracy for the results. Throughout the experimentation process, the time is reported on average by running it ten times, whereas the positioning accuracy for the analysis of a predicted position (longitude and latitude) per test fingerprint uses the *Similarity* function (in meters) in (38) based on the *Euclidean* distance. To analyze N test fingerprints from the validation database, three accuracy metrics (in meters) are defined for evaluating the performance of the results. First, the *Error* metric in (36) considers the two dimensional (2D) mean positioning accuracy of the test fingerprints regardless the correctness of building and floor identification. On the other hand, the $Error^*$ metric considers the 2D mean positioning accuracy of the test fingerprints whose building and floor are correctly matched. Last, the $Error^{pn}$ metric in (37) considers the 2D mean positioning accuracy of the test

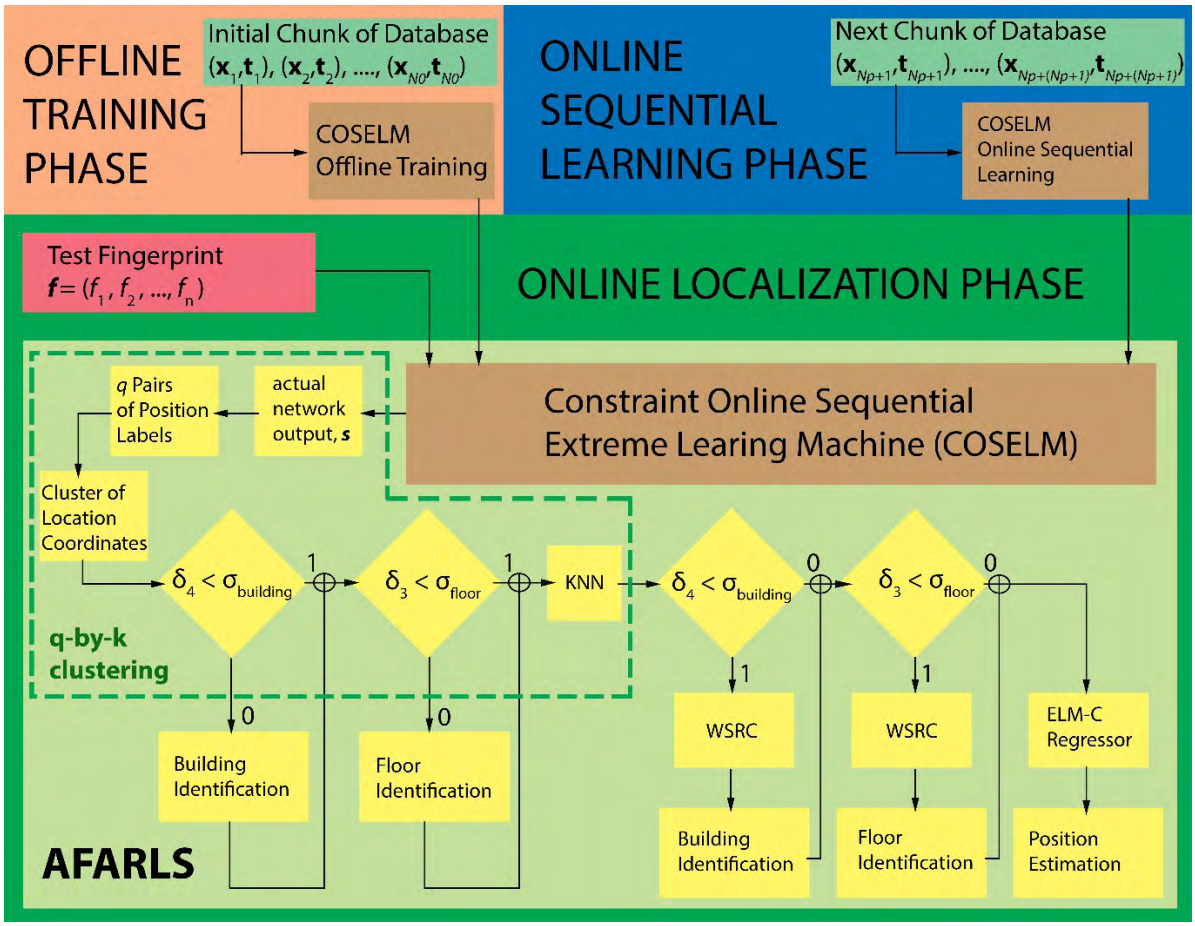

FIGURE 1. Overview of AFARLS.

fingerprints by including the building and floor estimation penalties based on the competition rule at EvAAL on IPIN 2015 [35].

$$Error = \frac{1}{N}\sum_{i=1}^{N} Similarity_i\,(\boldsymbol{x},\boldsymbol{y}) \tag{36}$$

$$Error^{pn} = \frac{1}{N}\sum_{i=1}^{N} Similarity_i\,(\boldsymbol{x},\boldsymbol{y}) + P_1 \times B_i + P_2 \times F_i \tag{37}$$

$$Similarity_i\,(\boldsymbol{x},\boldsymbol{y}) = \sqrt{(x_i-\tilde{x}_i)^2 + (y_i-\tilde{y}_i)^2} \tag{38}$$

where the $Similarity_i$ is based on the *Euclidean* distance metric, (x_i, y_i) and $(\tilde{x}_i, \tilde{y}_i)$ is the true and the estimated positions of the i^{th} test fingerprint, P_1 and P_2 are the penalties associated to wrong building and floor estimations, B_i is 1 for wrong building estimations and 0 otherwise, F_i is the absolute difference between the true floor and the estimated floor. P_1 and P_2 are set to 50 and 4 *meters* respectively. To evaluate the classification accuracy in predicting the building and the floor successfully, we define *hit rate* and *success rate*. The *building hit rate* corresponds to the percentage of the test fingerprints whose building is correctly predicted. The *floor success rate* corresponds to the percentage of the test fingerprints whose building and floor are correctly predicted. If we ignore the correctness of the building identification, the *floor success rate* becomes the *floor hit rate* that corresponds to the percentage of the test fingerprints whose floor is correctly predicted. Hence, we consider the best localization metric is the one in which its classification rate (highest *hit rate* or *success rate*) is the highest, and then followed by the highest mean positioning accuracy (lowest *Error*, *Error** or $Error^{pn}$).

A. EXPERIMENT 1: SELECTION OF PARAMETERS FOR THE COSELM CLASSIFIER IN AFARLS

Since COSELM is the crucial factor of our proposed model to determine the overall performance, we investigate the critical parameters that can affect its performance in this section with the limited training fingerprints in the positive data representation based on the *EU Zenodo* database. We start by training the classifier to identify the floor, while training the regressor to estimate the position. Both the classifier and the regressor are based on ELM. After that, the performance of the classification rate and the positioning accuracy using four different activation functions are evaluated. These activation functions

Algorithm 6 AFARLS

Input: Fingerprint database, $D = \{(\boldsymbol{x}_j, \boldsymbol{t}_j)|\boldsymbol{x}_j \in \boldsymbol{R}^n, \boldsymbol{t}_j \in \boldsymbol{R}^m\}_{j=1}^{N}$, where $m = 4$ for multi-label classification, activation function $G(\cdot)$, hidden node parameters (L_{COSELM}, L_{ELM-C}), initial chunk of data size N_0, subsequent chunks of online incremental training data size N_{p+1}, clustering parameters (q, k), threshold parameters $(\sigma_{building}, \sigma_{floor})$, test fingerprint $\boldsymbol{f}$.
Output: Predicted Coordinates, $\boldsymbol{Z} \in \boldsymbol{R}^m$ that considers building, floor and position (longitude, latitude) as a location
Pre-processing Phase:
1: Round off reference locations, $\boldsymbol{t}$, to the nearest integers, return $\boldsymbol{t}$
2: Represent non-heard and very weak-signal AP fingerprints in $\boldsymbol{x}$ with 0
3: Convert $\boldsymbol{x}$ to positive integers based on (31), return $\boldsymbol{x}$
Offline Training Phase:
1: With $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)|\boldsymbol{x}_i \in \boldsymbol{R}^n, \boldsymbol{t}_i \in \boldsymbol{R}^m\}_{i=1}^{N_0}$, train the COSELM model based on algorithm 3
Online Sequential Learning Phase:
if the $(p+1)^{th}$ chunk of incremental data arrives, **then**
1: Update the COSELM model, with $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)|\boldsymbol{x}_i \in \boldsymbol{R}^n, \boldsymbol{t}_i \in \boldsymbol{R}^m\}_{i=N_p+1}^{N_p+N_{p+1}}$
end if
Online Localization Phase:
1: Generate $\boldsymbol{s}$ from the COSELM model, given $\boldsymbol{f}$
2: Perform *q-by-k clustering*
2.1: Cluster q pairs of position labels corresponding to the respective top q confident scores in $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$
2.2: Establish a cluster of coordinates whereby either their longitude or latitude labels match to these labels
2.3: **if** $\delta_4 \geq \sigma_{building}$ **then**
a: Identify the building, Z_4 by solving $\text{Label}(\boldsymbol{f}) = \underset{c_4 \in \{1,2,\ldots,M_4\}}{\arg\max} (\boldsymbol{s}_4)$
b: Filter the cluster in which the building label of the coordinates is matched with Z_4
2.4: **end if**
2.5: **if** $\delta_3 \geq \sigma_{floor}$ **then**
a: Identify the floor, Z_3 by solving $\text{Label}(\boldsymbol{f}) = \underset{c_3 \in \{1,2,\ldots,M_3\}}{\arg\max} (\boldsymbol{s}_3)$
b: Filter the cluster in which the floor label of the coordinates is matched with Z_3
2.6: **end if**
2.7: Extract $\boldsymbol{p}$ associated to these coordinates
2.8: Convert $\boldsymbol{p}$ to the *powed* data representation based on (32), return $\hat{\boldsymbol{p}}$
2.9: Determine $\boldsymbol{o}$ using KNN in algorithm 1 based on the *Sorensen* distance metric
3: **if** $\delta_4 < \sigma_{building}$ **then**
a: Identify the building, Z_4 with $\boldsymbol{o}$ based on WSRC in algorithm 5
b: Return $\boldsymbol{o}$ in which the corresponding building label is matched with Z_4
4: **end if**
5: **if** $\delta_3 < \sigma_{floor}$ **then**
a: Identify the floor, Z_3 with $\boldsymbol{o}$ based on WSRC in algorithm 5
b: Return $\boldsymbol{o}$ in which the corresponding floor label is matched with Z_3
6: **end if**
7: Use the ELM-C regressor in algorithm 2 with $\boldsymbol{o}$ to estimate the position (Z_1, Z_2)
8: Return $\boldsymbol{Z} = [Z_1, Z_2, Z_3, Z_4]$
9: **end**

are the sigmoid function (*sig*), radial basis function (*rbf*), sine function (*sin*) and hard-limit transfer function (*hardlim*).

Referring to Table 1, the top two best performances presented by the *sig* function and the *hardlim* function show that they are more suitable to accept the raw data in positive data representation. On the contrary, the performance of the RBF is the worst. Since the performance and the training speed using the *sig* function is better than the *hardlim* function, it is selected as the activation function for the ELM approach in IPS. Next, the original ELM is developed to ELM-C and COSELM. Comparison of their performances is carried out with the same activation function, and the initial number of hidden nodes, L is selected to be 200. According to the results in Table 2, the original ELM performs poorly compared to the others. The performance improves substantially after introducing the optimal parameter λ_{opt} into ELM, referred to as ELM-C. Besides the activation function, the number of the hidden nodes is another key parameter to determine the performance of the positioning accuracy using the ELM approach. L in the range of 200 to 5000 is studied for its effect

TABLE 1. Selection of the type of activation function for ELM on the *EU Zenodo* database.

Approaches	Activation Function, $G(\cdot)$	Number of Hidden Neurons, L	Offline Training Phase			Online Testing Phase		
			Time, s	Floor Hit Rate, (%)	Error*, m	Time, s	Floor Hit Rate, (%)	Error*, m
ELM	sig	200	**0.10**	**97.99**	**11.93**	0.13	**88.16**	**16.49**
ELM	rbf	200	5.40	2.58	61.98	29.46	2.05	99.78
ELM	sin	200	0.24	63.56	75.60	0.14	23.24	106.38
ELM	hardlim	200	0.26	97.70	12.08	**0.11**	87.90	16.59

TABLE 2. Performance comparisons on the *EU Zenodo* database between ELM with and without λ and different number of L.

Approaches	Activation Function, $G(\cdot)$	Number of Hidden Neurons, L	Offline Training Phase			Online Testing Phase		
			Time, s	Floor Hit Rate, (%)	Error*, m	Time, s	Floor Hit Rate, (%)	Error*, m
ELM	sig	200	**0.10**	97.99	11.93	**0.13**	88.16	16.49
ELM-C	sig	200	0.52	95.27	12.29	**0.13**	89.47	14.76
ELM-C	sig	1000	2.15	99.43	5.51	0.55	94.18	11.46
ELM-C	sig	3000	6.80	99.86	2.12	1.59	94.36	9.81
ELM-C	sig	5000	21.42	**100.00**	**1.41**	2.81	**94.48**	**9.27**

TABLE 3. Effects of the parameters, L and (N_0, N_{p+1}) on the ELM performance on the *EU Zenodo* database.

Approaches	Number of Calibration Points (N_0, N_{p+1})	Number of Hidden Neurons, L	Offline Training Phase			Online Testing Phase		
			Time, s	Floor Hit Rate, (%)	Error*, m	Time, s	Floor Hit Rate, (%)	Error*, m
COSELM	697+0	200	0.52	95.27	12.29	**0.13**	89.47	14.76
COSELM	400+50	200	0.34	95.41	12.20	**0.13**	89.72	14.81
COSELM	200+50	200	**0.31**	95.41	12.10	**0.13**	89.60	14.90
COSELM	697+0	1000	2.32	99.43	5.51	0.59	94.18	**11.46**
COSELM	400+50	1000	3.26	99.43	**4.58**	0.50	**94.20**	11.92
COSELM	200+50	1000	3.34	**99.57**	5.31	0.52	94.03	11.56

on the ELM-C performance. As observed in Table 2, the training time of ELM-C increases dramatically when the size is 1000 onwards, incremented with a step size of 2000. Despite of the increase in time computation, both the training and testing accuracies become relatively stable when the size increases to 1000. In general, higher values of L generate more stable and better performance, but it sacrifices its learning speed and it costs more time in the testing phase. Since we will combine ELM with other algorithms for improvement and consider the trade-off between the time and the accuracy, we select the parameter $L = 1000$ as the optimal condition for the experiments since it can generate the results very close to the best within the acceptable computation time. Introducing online sequential learning method into ELM-C, it becomes COSELM. The performance is evaluated by splitting the 697 training samples into different pairs of (N_0, N_{p+1}), together with the relationship between N_0 and L. As one can see in Table 3, the *floor hit rate* is the best when we select $N_0 = 400$, $N_{p+1} = 50$ and $L = 1000$, while the testing *Error** reduces to 11-12m when $N_0 < L$. The significant increase in the *floor hit rate* and the positioning accuracy can be explained when we increase L larger than N_0, but it costs more in the computational time. Meanwhile, it is noted that the sequential learning method can speed up the learning process only when $N_0 \geq L$. Therefore, the appropriate selection of L and N_0 is crucial for the accuracy performance and the time complexity tradeoff. With a larger training database, the online sequential learning method becomes more essential to help in reducing the training time. We can realize its noteworthy speed advantage in Experiment 3 when we utilize a bigger training database and set $N_0 \geq L$.

B. EXPERIMENT 2: PERFORMANCE EVALUATION BASED ON THE EU ZENODO DATABASE

The design parameters in AFARLS are configured as follows on the *EU Zenodo* database, unless stated otherwise. *Sig* is selected as the activation function, σ_{floor} is 0.2, the number of hidden nodes of COSELM and ELM-C (L_{COSELM}, L_{ELM-C}) are 1000 and 100 respectively, the clustering parameter pair (q, k) are 8 and 4 respectively, and τ is set to $-103dBm$. The training dataset of 697 samples is split into the initial data chunk size N_0 of 400, followed by the subsequent multiple data chunks with the size N_{p+1} of 50.

The performance comparisons between AFARLS and the benchmark positioning algorithms on the *EU Zenodo* database is reported in Table 4. In this table, we see AFARLS

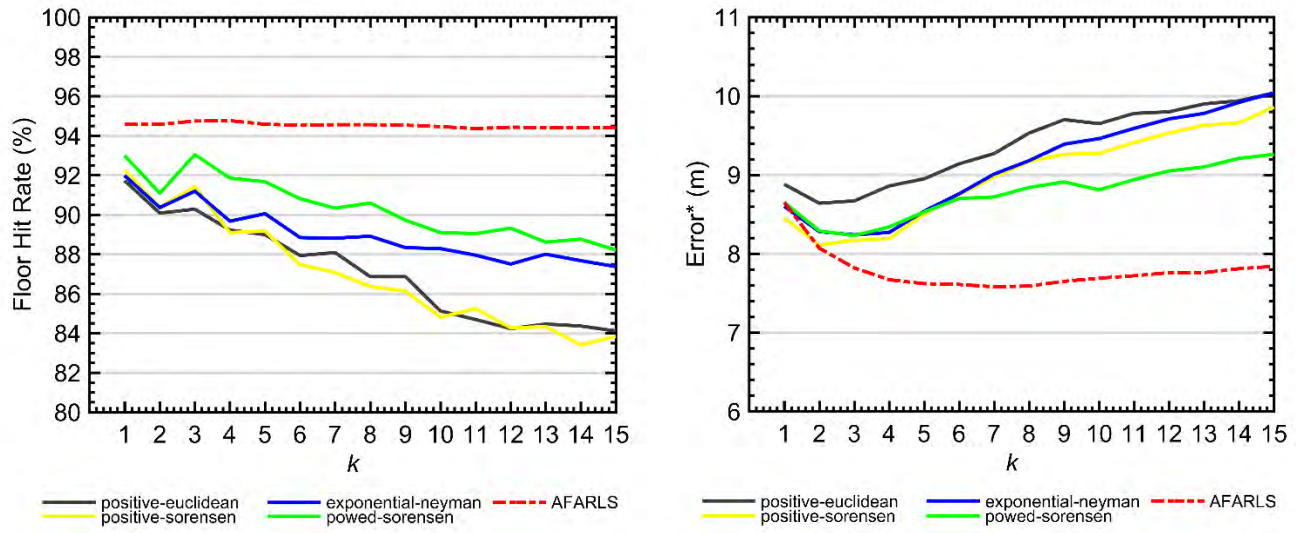

FIGURE 2. Performance comparisons between AFARLS and the KNN approaches on the EU Zenodo database during online testing phase. (a) Floor Hit Rate vs k Nearest Neighbors. (b) Erro r^* vs k Nearest Neighbors.

outperforms other algorithms. In the meantime, we list the best results from different approaches of KNN with their respective k compared with AFARLS in Table 5. As shown in the table, AFARLS presents the best performance by utilizing less computational time compared to the best result among all KNN approaches. The outstanding performance of AFALS in a single-building indoor environment is because of it has KNN and WSRC, which work together to further enhance the COSELM results. We also test the performance of AFARLS with and without the help of KNN and WSRC. Shown in the same table, AFARLS uses only COSELM and skips the algorithms of WSRC and KNN when $q = 0$ and $k = 0$. As a result, despite of it reduces the testing time significantly, its performance drops heavily. It is also noted that the training time of AFARLS remains approximately the same with and without the sequential learning method; this is because the training database is not very huge, whereby N_0 is smaller than L. Different adaptive sub-dictionary sizes which contain different numbers of nearest neighbors, k and $q = 8$ are studied to evaluate their performances. Their performances are compared with the KNN approaches by varying k from 1 to 15. Figure 2(a) relates the effect of k to *floor hit rate* in AFARLS and compares AFARLS with the performances using KNN approaches. On the other hand, Figure 2(b) describes the effect of the number of k to testing *Error**. As depicted in both figures, AFARLS trumps the KNN algorithms regardless of the factor of k, and it generates the results very fast for the real-time applications.

C. EXPERIMENT 3: PERFORMANCE EVALUATION BASED ON THE UJIIndoorLoc DATABASE

The design parameters in AFARLS are configured as follows on the *UJIIndoorLoc* database, unless stated otherwise. *Sig* is selected as the activation function, $\sigma_{building}$ is 0.2, σ_{floor} is 0.8, the number of hidden nodes of COSELM and ELM-C (L_{COSELM}, L_{ELM-C}) are 1000 and 100 respectively, the clustering parameter pair (q, k) are 8 and 6 respectively, and τ is set to $-104dBm$. The training dataset of 19937 samples is split into the initial data chunk size N_0 of 2000, followed by the subsequent multiple data chunks with the size N_{p+1} of 500.

TABLE 4. Performance comparisons between AFARLS and the benchmark positioning approaches on the *EU ZENODO* database.

Approaches	Floor Hit Rate, (%)	Testing Error*, m
Weighted Centroid [36]	83.19	10.64
3D clustering [37] (k-means)	72.90	17.35
RSS clustering [37] (affinity propagation)	90.81	8.09
Log-Gaussian Probability [37, 38] ($\sigma = 7$, $k = 1$)	85.29	9.78
RTLS@UM [35, 39] (*approach*=1, *variant*=1, n=k1=5, k2=3)	86.99	9.18
RTLS@UM [35, 39] (*approach*=1, *variant*=3, n=5, k1=1, k2=3)	90.05	9.18
AFARLS ($N_0 = 400, N_{p+1} = 50$, q=8, k=4)	**94.76**	**7.58**

Table 6 reports the results from the recent existing benchmark positioning algorithms on the *UJIIndoorLoc* database. As discussed before, the testing data is only provided to the competitors at EvAAL. Therefore, a direct performance comparison between AFARLS and the first four positioning algorithms from the EvAAL/IPIN 2015 competition is not

TABLE 5. Performance comparisons between AFARLS with the KNN approaches on the *EU ZENODO* database.

Approaches	Offline Training Phase	Online Testing Phase			
	Time, s	Time, s	Floor Hit Rate, (%)	Error, m	Error*, m
UJI KNN Algorithm [16] (data=positive, dist=Euclidean, k=1)	-	17.52	91.72	9.30	8.88
UJI KNN Algorithm [16] (data=positive, dist=Sorensen, k=1)	-	32.56	92.26	8.64	8.45
UJI KNN Algorithm [16] (data=exponential, dist=neyman, k=1)	-	61.59	91.98	8.92	8.60
UJI KNN Algorithm [16] (data=powed, dist=Sorensen, k=3)	-	34.23	93.04	8.57	8.23
AFARLS ($N_0 = 697, N_{p+1} = 0$, q=0, k=0)	2.40	**0.57**	94.18	10.07	9.85
AFARLS ($N_0 = 697, N_{p+1} = 0$, q=8, k=4)	2.42	21.72	94.74	**7.96**	**7.58**
AFARLS ($N_0 = 400, N_{p+1} = 50$, q=8, k=4)	2.42	22.02	**94.76**	**7.96**	**7.58**

TABLE 6. Performance comparisons between AFARLS and the benchmark positioning approaches on the *UJIIndoorLoc* database.

Approaches	Building Hit Rate, (%)	Floor Hit Rate, (%)	Testing Error^{pn}, m
MOSAIC [35]	98.65	93.86	11.64
HFTS [35]	100	96.25	8.49
RTLS@UM [35]	100	93.74	6.20
ICSL [35]	100	86.93	7.67
DNN [40] (k=8, 0.2)	99.82	91.27	9.29
Deep Learning [41]	92	92	-
AFARLS	100	95.41	6.40

possible. On the other hand, the fifth positioning algorithm validates the result by splitting the training data into new training and validation sets with the ratio of 70:30. The last positioning algorithm adopts the performance evaluation method similar to ours by treating the *UJIIndoorLoc* validation data as the testing samples. Even though the direct comparison is not allowed, our results are very comparable to the results presented in Table 6. Meanwhile, we list the best results from different approaches of KNN with their respective k compared with AFARLS in Table 7. Summarized in the table, AFARLS makes the best classification performances in *building hit rate* and *floor success rate* with the help of WSRC and KNN, whereas its testing *Error** is the second-lowest. Furthermore, it is noteworthy that the testing time of AFARLS is reported to be the lowest. This is because AFARLS benefits from the speed advantage of ELM. With the help of the online sequential learning algorithm, the training time can also be reduced to almost half. The same as with Experiment 2, we also test the performance of AFARLS with and without the help of KNN and WSRC. When $q = 0$ and $k = 0$, AFARLS utilizes only COSELM and skips the algorithms of WSRC and KNN. As expected in Table 7, the performance of AFARLS drops heavily using only COSELM, but the testing time has been saved significantly. To validate the effect of different number of nearest neighbors k to the performance in AFARLS and to compare it with the performances using the KNN approaches, we vary k from 1 to 15 and set $q = 8$. Figure 3(a) and Figure 3(b) describe the effect of k on *building hit rate* and *floor success rate* respectively, while Figure 3(c) depicts the effect of k on the testing *Error**. From these figures, we see that AFARLS outperforms many KNN algorithms in classifying the building and the floor, while the position estimated from AFARLS is around 0.2-0.3m different compared to the best performance among all KNN algorithms. Nevertheless, the unique advantage of AFARLS over KNN is that it can reduce significantly the cost of online testing time. Once trained, it can estimate the locations faster than KNN given the test fingerprint database because it benefits from the speed advantage of the ELM algorithm. In addition, the online sequential learning method introduced to ELM helps AFARLS in speeding up the offline training process especially when we deploy it in a larger environment with a bigger training database. As a result, AFARLS saves remarkable training and testing time.

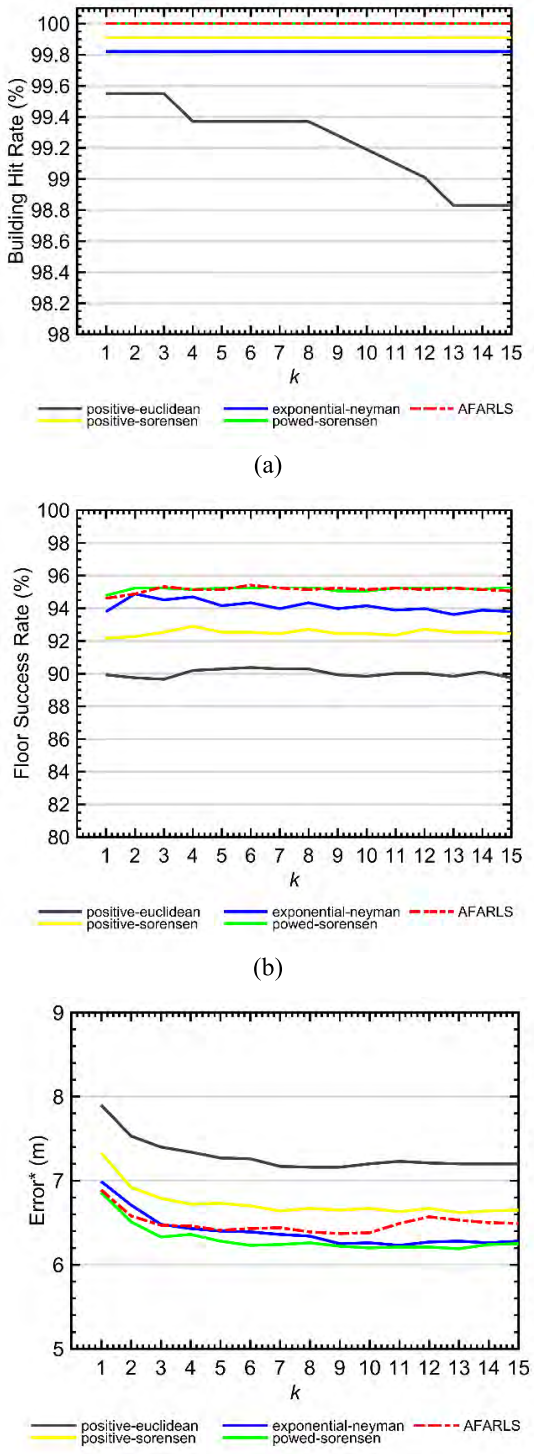

FIGURE 3. Performance comparisons between AFARLS and KNN on the *UJIIndoorLoc* database during online testing phase. (a) Building Hit Rate vs *k* Nearest Neighbors. (b) Floor Success Rate vs *k* Nearest Neighbors. (c) Error* vs *k* Nearest Neighbors.

Apart from being fast in the training and testing phases, AFARLS can update itself by using the online incremental data so that it can adapt itself to a new and different environment without retraining the whole system. To show this feature with the *UJIIndoorLoc* dataset, we sort its training and test samples according to the building label. Hence, we have three sets of training datasets and three sets of test datasets. The first set of training and test datasets corresponds to the samples that belong to the first building, while the second and third datasets correspond to the samples that belong to the second and third buildings respectively. During the offline training phase, we establish an initial AFARLS model with the first training dataset. After that, we evaluate the system performance by using the testing samples that belong to the first test dataset. Next, we update the existing model during online sequential learning phase by using the subsequent chunk of dataset in which all the training samples belong to the second training dataset. Then, we evaluate the system performance with the first and the second test datasets. Last, we update the model again by using the third dataset and evaluate the system performance by using all the test datasets.

As illustrated in Table 8, AFARLS performs very well as expected in a single building environment. This is because the test bed is small. When the environment is enlarged to include additional new buildings (the second and third buildings), the performance declines slightly. It should also be noted that these results are very similar to the results we have obtained in the previous section when we train the system with all the training data fully prepared in advance during the offline training phase. In fact, preparing all the data in advance is labor-cost and time-consuming. It is very hard to collect all the required training data ahead of time. Therefore, AFALRS makes use of the online sequential learning method to suit the way the new training data arrives without sacrificing the accuracy and it adapts itself to a new environment without retraining a new model with all the training data.

To show the impact of the online incremental data on the system performance, we randomly select 10000 samples from the training dataset as the online incremental data to reflect the environmental dynamics. In other words, we set up an initial model in AFARLS with the initial training data of 9937 samples during offline training phase. During the online sequential learning phase, we update the existing model with five chunks of datasets sequentially in which each dataset has 2000 incremental data. Then, we evaluate the performance of AFARLS after every update to the model with a step size of 2000 samples. Figure 4(a) depicts the influence of the online incremental data to the system performance in the online localization phase, after every update to the model. As shown, the initial localization performance is the worst. After gradually refining the model with the online incremental data, the performance improves because the newly updated model can reflect the current indoor environment better. This tells us that AFARLS can improve and update itself along the timeline by utilizing the incremental learning method for a lifelong and high-performance running.

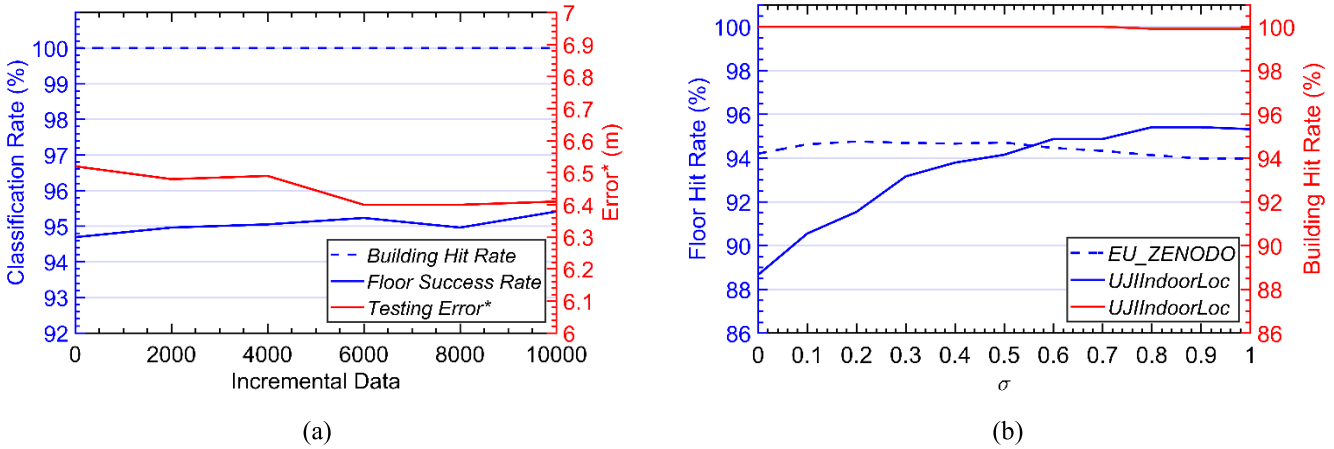

FIGURE 4. (a) Performance evaluation of AFARLS with respect to multiple chunks of online incremental data on the *UJIIndoorLoc* database. (b) Misclassification comparisons of AFARLS with respect to different σ on the *EU Zenodo* and the *UJIIndoorLoc* databases.

TABLE 7. Performance comparisons between AFARLS and the KNN approaches on the *UJIIndoorLoc* database.

Approaches	Offline Training Phase	Online Testing Phase				
	Time, *s*	Time, *s*	Building Hit Rate, (%)	Floor Success Rate, (%)	Error, *m*	Error*, *m*
UJI KNN Algorithm (data=positive, dist=euclidean, k=1)	-	84.01	99.55	89.92	9.63	7.90
UJI KNN Algorithm (data=positive, dist=Sorensen, k=4)	-	164.00	99.91	92.89	7.37	6.72
UJI KNN Algorithm (data=exponential, dist=neyman, k=2)	-	230.50	99.82	94.87	7.39	6.71
UJI KNN Algorithm (data=powed, dist=Sorensen, k=7)	-	148.36	**100**	95.32	**6.58**	**6.24**
AFARLS ($N_0 = 19937, N_{p+1} = 0$, q=0, k=0)	86.07	**0.21**	**100**	88.66	18.41	16.47
AFARLS ($N_0 = 19937, N_{p+1} = 0$, q=8, k=6)	84.68	22.20	**100**	**95.41**	6.78	6.40
AFARLS ($N_0 = 2000, N_{p+1} = 500$, q=8, k=6)	40.02	22.20	**100**	**95.41**	6.78	6.40

D. EXPERIMENT 4: PERFORMANCE EVALUATION OF AFARLS WITH DIFFERENT THREDHOLDS AND PARAMETERS

In this experiment, the threshold σ is tested on the grid [0:0.1:1]. It is to demonstrate the variations of *building hit rate* and *floor hit rate* after partitioning the test fingerprint database into subsets based on σ. For the test fingerprints classified as the noisy fingerprints, AFARLS incorporates the classification capability of WSRC to identify the floor. Following the analysis in Figure 4(b), the highest *floor hit rate* in the experiment based on the *EU ZENODO* database is achieved when $\sigma_{floor} = 0.2$, while the best *floor hit rate* in the experiment based on the *UJIIndoorLoc* database is achieved when $\sigma_{floor} = 0.8$. In addition, it can be realized that the result based on the former experiment does not have much gain corresponding to the change of σ, compared with the latter. It can be explained in such a way that the fingerprint database of the former experiment contains little noise when it is used to classify the floor, whereas the latter experiment carried out in the larger (multi-building) environment contains the noisier fingerprint database when it is used to classify the floor but the noise is distinguishable when the database is used to classify the building. It can be seen from Figure 4(b) that the *building hit rate* is stable for $\sigma_{building} \in [0, 1.0]$.

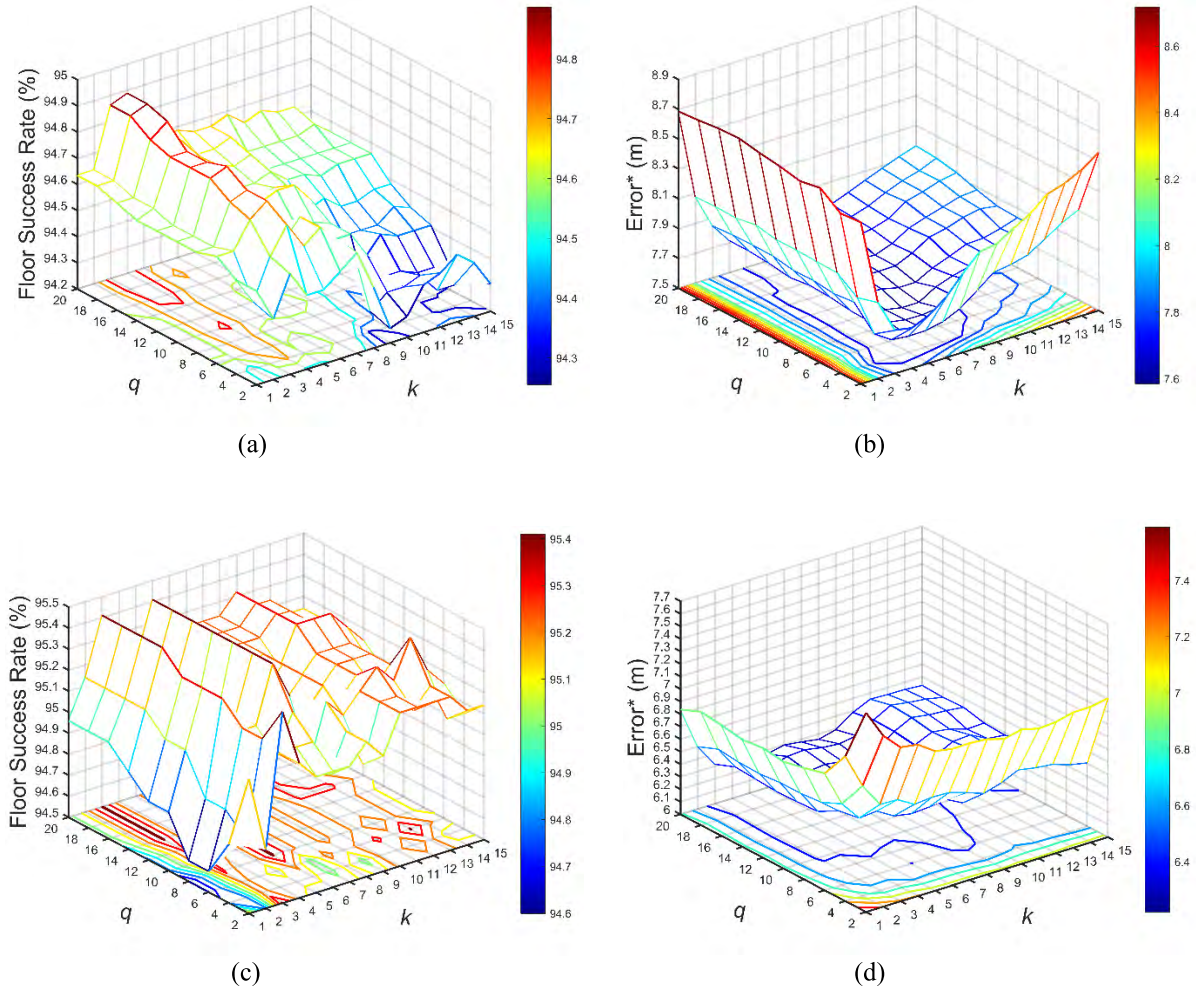

FIGURE 5. Misclassification and testing accuracy comparisons of AFARLS with respect to various combinations of q and k during online testing phase. Tested on the *EU Zenodo* database: (a) Floor Success Rate vs k Nearest Neighbors. (b) Error* vs k Nearest Neighbors. Tested on the *UJIIndoorLoc* database: (c) Floor Success Rate vs k Nearest Neighbors. (d) Error* vs k Nearest Neighbors.

TABLE 8. Performance evaluation of AFARLS with respect to three training *UJIIndoorLoc* datasets sorted according to the building label.

Testing Samples	Online Incremental Data	Building Hit Rate, (%)	Floor Success Rate, (%)	Error*, m
1st Building's Samples	-	100	97.76	4.77
1st & 2nd Buildings' Samples	2nd Buildings' Samples	100	94.90	5.91
1st, 2nd, & 3rd Buildings Samples	3rd Buildings' Samples	100	95.41	6.40

Therefore, we consider a small threshold $\sigma_{floor} = 0.2$ as the optimized condition in the former experiment, and utilize $\sigma_{building} = 0.2$ for the building identification, but adopt $\sigma_{floor} = 0.8$ which has a larger noise tolerance for the floor identification in the latter experiment. Next, we realize that the choice of the clustering parameter pair (q, k) determines the adaptive sub-dictionary size. Larger values of q and k produce bigger size of the sub-dictionaries, followed by a higher cost of computational complexity. Figure 5(a) and Figure 5(b) plot the testing results in the experiments based on the *EU ZENODO* database and Figure 5(c) and Figure 5(d) plot the testing results in the experiments based on the *UJIIndoorLoc* database with respect to various combinations of q and k, respectively. The *floor success rate* is reported as 94.59% on average, and the testing *Error** is reported as $7.74m$ on average in the regions of $q \in [8, 20]$ and $k \in [4, 15]$ on the *EU Zenodo* database, depicted in Figure 5(a) and Figure 5(b). On the other hand, the *floor success rate* is reported as 95.22% on average, and the testing *Error** is reported as $6.38m$ on average in the regions of $q \in [6, 20]$ and $k \in [3, 15]$ depicted in Figure 5(c) and Figure 5(d). It is worth pointing out that larger values of q and k do not guarantee better performance, but they will lead to a stable performance. For instance, we can see that the regions where $q \in [2, 4]$ and $k \in [1, 2]$ in both experiments produce very low accuracy results. The worst cases that happen in both experiments have

been discussed previously in Table 5 and 7 when $q = 0$ and $k = 0$, the results drop dramatically though the computation cost are extremely small. Hence, the proper selection of these parameters is vital for generating the appropriate size of the sub-dictionary that contains sufficient informative data to work out accurate results within the acceptable computational time. In view of all experiments above, AFARLS exploits the speed advantage of ELM to reduce the computational cost, and the accuracy advantage of SRC to enhance the classification performance, by utilizing KNN as the adaptive sub-dictionary selection strategy. The baseline results that are based on the traditional KNN are 91.72% for the *floor hit rate* and 8.88*m* for the testing *Error** on the *EU Zenodo* dataset. On the other hand, the baseline results that are based on the traditional KNN are 99.55% for the *building hit rate*, 89.92% for the *floor success rate* and 7.90*m* for the testing *Error** on the *UJIIndoorLoc* dataset. Compared to our proposed model, the results from AFARLS are 94.76% for the *floor hit rate* and 7.58*m* for the testing *Error** on the former dataset, while they are 100% for the *building hit rate*, 95.41% for the *floor success rate* and 6.40*m* for the testing *Error** on the latter dataset. Thus, AFARLS has enhanced the *floor hit rate* by 3.31% and the testing *Error** by 14.64% based on the former dataset, while the *building hit rate* is enhanced by 0.45%, the *floor success rate* by 6.11% and the testing *Error** by 18.99% based on the latter dataset. As one can see, the computational cost of the traditional KNN approach based on the latter dataset (i.e., 84.01*s*) is very large compared to the former dataset (i.e., 17.52*s*) during the online testing phase. The intensive computational cost in KNN is on account of the huge training sample size of the latter dataset. Unlike KNN, AFARLS demonstrates its outstanding computational efficiency via the adaptive sub-dictionary selection strategy. When the size of the dataset increases remarkably, AFARLS shows its testing time will not increase significantly and it can estimate the results faster than the KNN algorithms. More specifically, it saves up to almost 70% of the testing time compared to the traditional KNN approach based on the latter dataset. Not only is the testing speed improved, AFARLS also adopts the online sequential learning method to speed up the training speed. Apart from exploiting the online sequential learning ability for the speed advantage, AFARLS utilizes this method to update the existing model in a timely manner to environmental dynamics with online incremental data. Most importantly, AFARLS learns fast with a varying chunk size of the new incremental data. Without retraining a new model with all the training data to update the system, AFALRS cuts down the time consumptions and manpower costs for the site survey during the offline training phase. Considering these unique advantages, AFARLS outrivals the traditional fingerprinting-based positioning algorithms based on KNN.

V. CONCLUSION

In large-scale highly dynamic indoor environments, the outstanding WiFi-based IPS should offer advantages of not merely the high positioning accuracy, but also the lifelong performance running, together with the fast and feasible site survey. In this paper, we have proposed AFARLS to offer these advantages in the large-scale indoor environments and have validated the performance through extensive experimentation. The results show that AFARLS can offer real-time performance with high accuracy, and leverage online incremental data with a varying size to update the out-of-date model without retraining a new model. This novel adaptive Wi-Fi indoor localization model inherits the advantages from the original ELM, SRC and KNN algorithms in order to tackle their respective drawbacks which render their practical applications in IPS. Future improvement can be focused on filtering algorithms and more advanced ELM algorithms to enhance the performance and reduce the memory size of the reference database by storing only the reliable training fingerprints collected through the popular collaborative or crowd-sourced method.

ACKNOWLEDGMENT

The author would like to thank great appreciation towards MIMOS for the hardware assistance.

REFERENCES

[1] S. Ramnath, A. Javali, B. Narang, P. Mishra, and S. K. Routray, "IoT based localization and tracking," in *Proc. Int. Conf. IoT Appl. (ICIOT)*, 2017, pp. 1–4.

[2] SGM Caspari, "Global positioning systems in combat," U.S. Army Sergeants Major Acad., Fort Bliss, TX, USA, Tech. Rep. Accessed: Apr. 22, 2018. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/a596592.pdf

[3] S. He and S.-H. G. Chan, "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 466–490, 1st Quart., 2015.

[4] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales, and J. Fangmeyer, Jr., "Evolution of indoor positioning technologies: A survey," *J. Sensors*, vol. 2017, pp. 1–21, Mar.2017, Art. no. 2630413.

[5] R. Henniges, "Current approaches of Wi-Fi positioning," in *Proc. IEEE Conf. Publications*, 2012, pp. 1–8. Accessed: Apr. 22, 2018. [Online]. Available: https://www.snet.tu-berlin.de/fileadmin/fg220/courses/WS1112/snet-project/wifi-positioning_henniges.pdf

[6] S. Xia, Y. Liu, G. Yuan, M. Zhu, and Z. Wang, "Indoor fingerprint positioning based on Wi-Fi: An overview," *ISPRS Int. J. Geo-Inf.*, vol. 6, no. 5, p. 135, 2017.

[7] R. Kaune, "Accuracy studies for TDOA and TOA localization," in *Proc. 15th Int. Conf. Inf. Fusion (FUSION)*, 2012, pp. 408–415.

[8] J. Xu, M. Ma, and C. L. Law, "Cooperative angle-of-arrival position localization," *Measurement*, vol. 59, pp. 302–313, Jan. 2015.

[9] Y. Yan, H. Wang, X. Shen, K. He, and X. Zhong, "TDOA-based source collaborative localization via semidefinite relaxation in sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 9, p. 248970, 2015.

[10] G. Retscher, "Fusion of location fingerprinting and trilateration based on the example of differential Wi-Fi positioning," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 4, p. 377, Sep. 2017.

[11] J. Zhang, G. Han, N. Sun, and L. Shu, "Path-loss-based fingerprint localization approach for location-based services in indoor environments," *IEEE Access*, vol. 5, pp. 13756–13769, 2017.

[12] R. Javaid, R. Qureshi, and R. N. Enam, "RSSI based node localization using trilateration in wireless sensor network," *Bahria Univ. J. Inf. Commun. Technol.*, vol. 8, p. 58, Dec. 2015.

[13] G. Félix, M. Siller, and E. N. Álvarez, "A fingerprinting indoor localization algorithm based deep learning," in *Proc. 8th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, 2016, pp. 1006–1011.

[14] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional neural networks based indoor Wi-Fi localization using channel state information," *IEEE Access*, vol. 5, pp. 18066–18074, 2017.

[15] F. Ebner, T. Fetzer, F. Deinzer, and M. Grzegorzek, "On Wi-Fi model optimizations for smartphone-based indoor localization," *ISPRS Int. J. Geo-Inf.*, vol. 6, no. 8, p. 233, 2017.
[16] J. Torres-Sospedra, R. Montoliu, S. Trilles, Ó. Belmonte, and J. Huerta, "Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems," *Expert Syst. Appl.*, vol. 42, pp. 9263–9278, Dec. 2015.
[17] M. Zhao and J. Chen, "Improvement and comparison of weighted K nearest neighbors classifiers for model selection," *J. Softw. Eng.*, vol. 10, no. 1, pp. 109–118, 2016.
[18] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.
[19] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Netw.*, vol. 61, pp. 32–48, Jan. 2015.
[20] J. Liu, Y. Chen, M. Liu, and Z. Zhao, "SELM: Semi-supervised ELM with application in sparse calibrated location estimation," *Neurocomputing*, vol. 74, no. 16, pp. 2566–2572, 2011.
[21] X. Jiang, Y. Chen, J. Liu, Y. Gu, and L. Hu, "FSELM: Fusion semi-supervised extreme learning machine for indoor localization with Wi-Fi and Bluetooth fingerprints," *Soft Comput.*, vol. 22, no. 11, pp. 3621–3635, 2018.
[22] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[23] H. Zou, X. Lu, H. Jiang, and L. Xie, "A fast and precise indoor localization algorithm based on an Online sequential extreme learning machine," *Sensors*, vol. 15, no. 1, pp. 1804–1824, Jan. 2015.
[24] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.
[25] Y. Gu, J. Liu, Y. Chen, and X. Jiang, "Constraint online sequential extreme learning machine for lifelong indoor localization system," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2014, pp. 732–738.
[26] Y.-C. Chen, J.-R. Chiang, H.-H. Chu, P. Huang, and A. W. Tsui, "Sensor-assisted Wi-Fi indoor location system for adapting to environmental dynamics," in *Proc. 8th ACM Int. Symp. Modeling, Anal. Simulation Wireless Mobile Syst.*, 2005, pp. 118–125.
[27] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for WLAN location fingerprinting," in *Proc. 1st Annu. Int. Conf. Mobile Ubiquitous Syst., Netw. Services (MOBIQUITOUS)*, 2004, pp. 14–23.
[28] J. Cao, K. Zhang, M. Luo, C. Yin, and X. Lai, "Extreme learning machine and adaptive sparse representation for image classification," *Neural Netw.*, vol. 81, pp. 91–102, Sep. 2016.
[29] E. S. Lohan, J. Torres-Sospedra, H. Leppäkoski, P. Richter, Z. Peng, and J. Huerta, "Wi-Fi crowdsourced fingerprinting dataset for indoor positioning," *Data*, vol. 2, no. 4, p. 32, 2017.
[30] J. Torres-Sospedra *et al.*, "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, 2014, pp. 261–270.
[31] M. Bicego and M. Loog, "Weighted K-nearest neighbor revisited," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, 2016, pp. 1642–1647.
[32] M. van Heeswijk and Y. Miche, "Binary/ternary extreme learning machines," *Neurocomputing*, vol. 149, pp. 187–197, Feb. 2015.
[33] B. Altintas and T. Serif, "Improving RSS-based indoor positioning algorithm via k-means clustering," in *Proc. 11th Eur. Wireless Conf.-Sustain. Wireless Technol. (Eur. Wireless)*, Apr. 2011, pp. 1–5.
[34] C.-W. Lee, T.-N. Lin, S.-H. Fang, and Y.-C. Chou, "A novel clustering-based approach of indoor location fingerprinting," in *Proc. IEEE 24th Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2013, pp. 3191–3196.
[35] A. Moreira, M. J. Nicolau, F. Meneses, and A. Costa, "Wi-Fi fingerprinting in the real world-RTLS@ UM at the EvAAL competition," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, 2015, pp. 1–10.
[36] A. Razavi, M. Valkama, and E.-S. Lohan. (2015). "K-means fingerprint clustering for low-complexity floor estimation in indoor mobile localization." [Online]. Available: https://arxiv.org/abs/1509.01600
[37] A. Cramariuc, H. Huttunen, and E. S. Lohan, "Clustering benefits in mobile-centric WiFi positioning in multi-floor buildings," in *Proc. Int. Conf. Localization GNSS (ICL-GNSS)*, 2016, pp. 1–6.
[38] S. Shrestha, J. Talvitie, and E. S. Lohan, "Deconvolution-based indoor localization with WLAN signals and unknown access point locations," in *Proc. Int. Conf. Localization GNSS (ICL-GNSS)*, 2013, pp. 1–6.
[39] J. Torres-Sospedra *et al.*, "A realistic evaluation of indoor positioning systems based on Wi-Fi fingerprinting: The 2015 EvAAL–ETRI competition," *J. Ambient Intell. Smart Environ.*, vol. 9, no. 2, pp. 263–279, 2017.
[40] K. S. Kim, S. Lee, and K. Huang, "A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting," *Big Data Anal.*, vol. 3, p. 4, Dec. 2018.
[41] M. Nowicki and J. Wietrzykowski, "Low-effort place recognition with WiFi fingerprints using deep learning," in *Proc. Int. Conf. Autom.*, 2017, pp. 575–584.

HENGYI GAN received the B.S. degree in electrical and electronic engineering from Universiti Teknologi PETRONAS (UTP), in 2017, where he is currently pursuing the M.S. degree. He is currently a Research Associate with the Department of Electrical and Electronic Engineering, UTP. His research interest includes indoor localization, wireless sensor networks, machine learning, and artificial intelligence.

MOHD HARIS BIN MD KHIR (M'08) was born in Kedah, Malaysia, in 1968. He received the B.Eng. degree in electrical and electronic engineering from Universiti Teknologi MARA, Selangor, Malaysia, in 1999, the M.Sc. degree in computer and systems engineering from Rensselaer Polytechnic Institute, NY, USA, in 2001, and the Ph.D. degree in systems engineering from Oakland University, MI, USA, in 2010. He joined Universiti Teknologi PETRONAS (UTP), in 1999, where he is currently an Associate Professor with the Electrical and Electronic Engineering Department. He held several positions at UTP such as the Deputy Head of Department and the Director of mission oriented research on nanotechnology. Most devices were fabricated using CMOS and MUMPS technologies. He has published three book chapters and more than 37 journals and 70 conference paper in the area of sensor, actuator, energy harvester, and sensor's application in the IoT. His research interest include micro/nano–electro mechanical systems sensors and actuator development.

GUNAWAN WITJAKSONO BIN DJASWADI received the B.S. *(magna cum laude)* and M.S. degrees in electrical engineering from Michigan Technological University, Houghton, MI, USA, in 1992 and 1994, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Wisconsin–Madison in 2002. From 1994 to 1996, he was with the National Aeronautics and Space Agency, Indonesia. In 2002, he joined Denselight Semiconductors Pte Ltd., Singapore, where he developed high-speed, long wavelength, and distributed feedback lasers. He was with Finisar Malaysia to develop uncooled and high-speed optical transceiver. He was with the Department of Electrical Engineering, University of Indonesia, from 2005 to 2007, before joining MIMOS when he held various key positions such as a Principal Researcher and the Director of Research and Sensor System Architect, until 2016. He is currently an Associate Professor with the Electrical and Electronics Department, Universiti Teknologi PETRONAS, where he is also an Indonesia-Chapter Professional Engineer.

NORDIN RAMLI (M'04–SM'13) received the B.Eng. degree in electrical engineering from Keio University, Japan, in 1999, and the M.Eng. and Ph.D. degrees in electronic engineering from the University of Electro-Communications, Japan, in 2005 and 2008, respectively. He was with Telekom Malaysia, Berhad, as a Network Engineer, from 1999 to 2008, and a Lecturer with Multimedia University, Malaysia, from 2008 to 2009. He is currently a Senior Staff Researcher with Wireless Network and Protocol Research, MIMOS Berhad, Malaysia. He is also a Solution Architect for the Internet of Things (IoT) and big data related project. He has authored or co-authored over 80 journals and conference papers, and has filed over 30 patents related to wireless communications which are pending with World Intellectual Property Organization and the Intellectual Property Corporation of Malaysia. His current research interests include cognitive radio, TV white space, space-time processing, equalization, adaptive array system and wireless mesh networking, the IoT, and big data. He has been appointed as a member of the Young Scientist Network of Malaysia Academy of Science, since 2014. He received the Top Research Scientist Malaysia, in 2018. He has been the Chair of the White Space Working Group, a technical standardization working group, Malaysia Technical Standard Forum, Berhad, since 2013, to study and promote the technology of white space communication in Malaysia. He is currently an Associate Editor of *IEICE Communication Express*. He is also a Registered Professional Engineer with the Board of Engineers, Malaysia.