IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 17, 2020, accepted December 20, 2020, date of publication January 8, 2021, date of current version January 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3049593

Adaptive Indoor Localization System for Large-Scale Area

TEERAPAT VONGSUTEERA AND KULTIDA ROJVIBOONCHAI, (Member, IEEE)
Chulalongkorn University Big Data Analytics and IoT Center (CUBIC), Department of Computer Engineering, Faculty of Engineering, Wireless Network and Future Internet Research Unit, Chulalongkorn University, Bangkok 10330, Thailand

Corresponding author: Kultida Rojviboonchai (kultida.r@chula.ac.th)

This work was supported in part by the 100th Anniversary Chulalongkorn University Fund for Doctoral Scholarship, in part by the 90th Anniversary Chulalongkorn University Fund (Ratchadaphiseksomphot Endowment Fund), and in part by the Wireless Network and Future Internet Research Unit, Ratchadaphiseksomphot Endowment Fund, Chulalongkorn University.

ABSTRACT Generally, fingerprint-based indoor localization works inefficiently when deployed in a large-scale area. This is because it consumes massive resources and takes long processing time for searching the exact location in the large fingerprint database. Moreover, the changing environment can degrade overall performance. To tackle these problems, we propose an adaptive indoor localization system for a large-scale area. Our system consists of three main parts. First, our area classification algorithm is the key to overcome the problem caused by the large-scale area. It identifies an area of the user's queries whether they are outdoor or located in a specific building. Specifically, the algorithm can filter out the queries sent from outdoor or out-of-scope areas. Then, the information of this part is sent to the next part. Second, our fingerprint-based indoor localization algorithm can utilize the information from the first part by searching only the fingerprint in the specific building. This can significantly reduce searching space and processing time in order to localize the exact location. Third, our missing-BSSID detector algorithm detects the missing Basic Service Set Identifiers (BSSIDs) in the incoming query and updates a sampling database. This part is for our system to quickly adapt to the changing environment. We evaluated and deployed our system in a large-scale exhibition including 37 multi-floor buildings, covering 486,000 m^2 and generating approximately 600,000 records of queries from users. In addition, we created a simulation to evaluate our system in the critically-changing environment. Our proposed system achieves high accuracy. More importantly, our area classification algorithm can significantly reduce the overall processing time compared to the previous work. Also, we showed that when applying our missing-BSSID detector algorithm to our system as well as other existing systems, the overall system performance can be significantly improved.

INDEX TERMS Area classification, fingerprint, indoor localization, indoor localization system, large-scale, Wi-Fi.

I. INTRODUCTION

Recently, there have been several techniques proposed for indoor localization. These techniques can be classified into two categories. The first category is the group of techniques that use Bluetooth [1]–[4], Zigbee [5], Radio Frequency Identification (RFID) [6]–[10], Acoustic Signal [11], Wi-Fi probe request [12], [13], Channel State Information (CSI) [14] and Signal to Noise Ratio (SNR) [15]. These techniques require installation of infrastructure in advance, leading to expensive cost.

The second category is the group of techniques that do not require the pre-installation of infrastructure [16]–[35]. In [16]–[19], embedded sensors in smartphones such as accelerometer and gyro sensor are applied to count user's steps and track user's direction. Although this technique does not require the pre-installation of infrastructure, it still suffers from a cumulative error problem. In [20], [21], Global System for Mobile (GSM) is utilized for indoor localization. This technique uses the existing cellular infrastructure and achieves high accuracy for localization. However, many smartphone models limit access to GSM data [20]. Thus, the smartphone models cannot utilize GSM for indoor localization. In [22]–[25], geomagnetic is applied for indoor localization. This technique measures disturbance of the Earth's magnetic field affected by steel elements in the building. However, the variation of steel object's position in the building can cause low accuracy for indoor localization.

The associate editor coordinating the review of this manuscript and approving it for publication was Gang Li.

Among techniques in this category, the most popular technique is Wi-Fi fingerprint.

The Wi-Fi fingerprint can provide high accuracy for both 2D and 3D localization [16], [26]–[35]. This technique uses the existing Wi-Fi infrastructure which is normally provided in public buildings. Furthermore, a Wi-Fi interface is embedded in users' smartphones. Thus, this technique can be deployed without installing additional infrastructure and users are not required to carry any special devices. Wi-Fi fingerprint has the assumption that, in each area, there exists a unique signal fingerprint. The process can be divided into two phases: a training phase and a localization phase. First, in the training phase, one needs to survey signal fingerprints in the target area and store those fingerprints into the database. Second, in the localization phase, the localization algorithm is needed in order to compare Wi-Fi scanning results with the information in the database. Then, the algorithm returns the matching position. However, when applying the Wi-Fi fingerprint technique in the large-scale area, such as a big exhibition or convention centers, it suffers from the following problems.

1) Long processing time: The target area needs to cover many floors and buildings. As the size of the target area increases, the number of signal fingerprints in the database increases. As a result, the fingerprint technique consumes massive resources and takes long processing time.

2) Waste of resources: In practice, users are able to query for their location from any location including indoor (say, within scope area) and outdoor (say, out-of-scope area). In the case of query from outdoor, the traditional fingerprint technique wastes resources and processing time for searching fingerprints in all of the database. This is because it cannot find the fingerprint which is similar to the query. Moreover, in some cases, it may return an incorrect indoor location because the query, which is actually from the outdoor location, is similar to the signal fingerprint at that indoor location.

3) Performance degradation due to the changing environment: In a large-scale area such as a big exhibition, Wi-Fi signal fingerprint may be changed occasionally due to many reasons such as removing and installing new Wi-Fi access points or hotspot. These reasons can significantly reduce accuracy of the Wi-Fi fingerprint technique.

Previously, an area classification algorithm [26], [36]–[38] has been proposed to classify an area of the user which can be indoor, outdoor [36], [37] or located in a specific building [26], [38]. These area classification algorithms can identify a user's area to reduce searching space and computation. However, they do not consider the effect of the changing environment. The detailed discussion can be found in Section II. Briefly, the changing environment is one of the most important issues when applying the area classification algorithms in a large-scale exhibition. This is because the users may carry Wi-Fi hotspot for sharing internet to other devices, new Wi-Fi access points may temporarily be installed for exhibition's purposes, or some Wi-Fi access points may disappear due to the closed offices.

In our previous work [27], we proposed DiffHit, the fingerprint-based floor localization algorithm. DiffHit provides high accuracy (up to 100% of accuracy) compared to the existing fingerprint-based algorithm [28]. In this article, we extend our previous work and propose a new adaptive indoor localization system for a large-scale area to tackle all of the three above-mentioned problems. Our system consists of three parts. The first part is a new area classification algorithm named *ExtHit*. It can identify an area of the user whether it is indoor or outdoor. *ExtHit* filters out the queries sent from outdoor (say, out-of-scope) location, so there is no exhaustive searching in all of the database. Moreover, *ExtHit* does not require an outdoor fingerprint which can increase manpower to collect fingerprints. It can reduce searching space and processing time in the large-scale database by classifying fingerprints into specific buildings. The second part is a fingerprint-based indoor localization algorithm named *InHit*. It uses the information from *ExtHit* to reduce searching space and processing time in order to localize the exact location inside the specified building. The third part is a missing-BSSID detector algorithm named *MissingHit*. It detects the missing BSSIDs in the incoming query and updates a sampling database. Thus, the sampling database or the fingerprint database can adapt itself to the changing environment. Our *ExtHit* is flexible. It not only works perfectly with our *InHit*, but it is also compatible with other fingerprint-based indoor localization algorithms. Moreover, our *MissingHit* is also compatible with other fingerprint-based area classification algorithms and fingerprint-based indoor localization algorithms. Our system has been tested in a real large-scale exhibition which includes 37 multi-floor buildings, covers more than 486,000 m^2 and generates more than 600,000 records of queries from users. The exhibition area is shown in Figure 1. The results show that *ExtHit* achieves high accuracy for identifying the user's location whether it is indoor, outdoor or in a specific building. Therefore, it can limit searching space, leading to significantly-reduced processing time. According to the experiment results, our *MissingHit* can significantly improve the accuracy of *ExtHit*, *InHit* and other existing area classification algorithms and indoor localization algorithms. Besides, *ExtHit* is robust to heterogeneity, i.e., heterogeneous mobile devices and critically-changing environment.

Our contribution in this article is fourfold. First, we propose the area classification algorithm named *ExtHit*. The algorithm consists of three modules as follows: (1) The unknown-BSSID filtering module removes an unknown-BSSID in the user's query because these BSSIDs can reduce performance of the overall system including accuracy and processing time. (2) The indoor/outdoor identification module can identify whether the user is indoor or outdoor. This module prevents the system from wasting resources for processing the queries which are sent from the outdoor or out-of-scope area. (3) The building identification module can identify which building the user is located with extremely-high accuracy. The Wi-Fi fingerprint based indoor

FIGURE 1. The exhibition area in Chulalongkorn University.

localization algorithms can utilize this module to reduce their searching space in order to localize the exact location. Second, we propose *InHit* as a new 3D indoor localization algorithm. *InHit* achieves up to 100% accuracy for floor localization and has average error distance 0.12 meters for localization. Third, we propose *MissingHit*, as a missing-BSSID detector algorithm. The algorithm detects the missing BSSIDs in an incoming query. It achieves very high accuracy in detection. The fingerprint based indoor localization systems can adapt their fingerprint database by integrating with *MissingHit*. Fourth, we propose the overall system architecture in which *ExtHit*, *InHit* and *MissingHit* work together for adaptive indoor localization. This can be used as a guideline to build up the indoor localization system for a large-scale area.

This article is organized as follows. Section II discusses the related works. Our system including the proposed architecture and the proposed algorithms are described in Section III. The performance evaluation is discussed in Section IV. The conclusion is elaborated in Section V.

II. RELATED WORKS

Several techniques have been proposed for indoor localization. Table 1 compares these techniques in many aspects. As can be seen in Table 1, Bluetooth, Zigbee, RFID, Acoustic signal and Wi-Fi probe request requires additional infrastructure installation. This leads to expensive cost for deployment. Although Geo-magnetic, GSM and the Wi-Fi fingerprint do not require infrastructure installation in advance, these techniques still require site survey to collect signal characteristics in the target area. This consumes time and manpower for signal site survey. In the case of availability in commercial smartphones, Zigbee and RFID are not available in most of the smartphone models. Thus, users who want to use these techniques are required to carry special devices for indoor localization. Also, it has been recently difficult to use GSM signals for indoor localization because many smartphone models limit access to GSM data [20]. In case of accuracy, Bluetooth, Zigbee, RFID, Acoustic signal, Geo-magnetic and Wi-Fi fingerprint can achieve a high level of accuracy ($<$ 2.5 m error). GSM and Wi-Fi probe request can achieve a medium level of accuracy ($<$ 5 m error). Embedded sensors

TABLE 1. Comparison of technology for indoor localization.

Technique	Infrastructure installation required (High cost)	Site survey required	Availability in commercial smartphone	Accuracy	Robustness to changing environments/noises
Bluetooth	✓	×	✓	High	High
Zigbee	✓	×	×	High	High
RFID	✓	×	×	High	High
Acoustic signal	✓	×	✓	High	Low
Wi-Fi probe request	✓	×	✓	Medium	High
CSI	✓	✓	×	High	High
SNR	✓	✓	×	High	High
Geo-magnetic	×	✓	✓	High	Low
Embedded sensors	×	×	✓	Low	High
GSM	×	✓	✓or ×	Medium	High
Wi-Fi fingerprint	×	✓	✓	High	High

can achieve a low level of accuracy because they suffer from the cumulative error problem. In case of robustness to the changing environment or noises, techniques using an acoustic signal can suffer from acoustic noise. The accuracy of techniques using geo-magnetic can be degraded due to the variation of steel object's position in the room or building. For instance, moving tables or chairs in the room can degrade localization accuracy.

Wi-Fi fingerprint is the most popular technique for applying to indoor localization [39]–[42] although it requires time and manpower for site survey. This is because Wi-Fi technology is available in commercial smartphones and the Wi-Fi signals are typically available in buildings. Using Wi-Fi for indoor localization does not require additional cost for infrastructure installation.

Moreover, Wi-Fi fingerprint can achieve high accuracy for indoor localization and a fingerprinting process can reduce the impact of the noises from the changing environments. The indoor localization based on Wi-Fi signal mainly relies on the signal fingerprints which are collected and stored in the database. The localization process is done by comparing the current signal with those signal fingerprints in the database.

Recently, there have been many approaches proposed for indoor localization [16], [27]–[35]. However, in the large-scale area, all of these approaches cannot work efficiently. This is because the processing time of the algorithms significantly increases as the number of signal fingerprints in the database increases. Moreover, they do not concern the queries from the out-of-scope area which usually occur and can decrease the performance of the system.

Horus [29] is an indoor localization system based on Wi-Fi signal fingerprint technique. In the offline phase, Horus creates radio map, clusters radio map and pre-processes

signal strength model. In the online phase, Horus performs four modules. First, the correlation handling module captures correlation between consecutive samples for obtaining a better location estimator. Second, the discrete space estimator module returns location which has maximum probability. Third, the small-scale compensator module handles small-scale variation of the wireless channel. Fourth, the continuous space estimator module returns a more exact location in continuous space by using the user's discrete estimated location as an input. However, Horus needs a long processing time when deployed in the large-scale area and the algorithm does not concern the queries from the out-of-scope area which can waste computational resources. Moreover, the accuracy can be degraded in the changing environment.

He *et al.* [26] proposed an area classification algorithm for large-scale area. The algorithm consists of three main modules: First, one-class inside/outside-region detection module classifies the outside signals as outliers. Second, an area classification module determines the area of the user for searching space reduction. Third, a device calibration module calibrates the device's scanning results. Nevertheless, this algorithm does not concern the impact of the changing environment which can occur when the system is deployed in a large-scale area. Thus, the overall performance of the algorithm can be degraded.

DiffHit [27] is a floor localization algorithm based on Wi-Fi signal fingerprint technique. The algorithm can be divided into two steps. First, the algorithm creates Wi-Fi signal fingerprints from Wi-Fi scanning results by using the top-N technique. Second, the algorithm utilizes the different order of Wi-Fi access points for comparing the current Wi-Fi signal with the Wi-Fi signal fingerprints in the database. The complexity of the algorithm is $O(n)$ where n is the number of fingerprints in the database. DiffHit was tested and compared with a previously-proposed indoor localization algorithm. It can achieve up to 100% accuracy. However, DiffHit suffers from long processing time when it is deployed in a large-scale area because the complexity of the algorithm increases as the number of fingerprints in the database increases.

RADAR [31] is an indoor localization system based on Wi-Fi fingerprint technique. The system calculates the similarity between the user's Wi-Fi measurement and Wi-Fi fingerprint in the database by using the Euclidean distance. Then, the system selects k fingerprints which provide the most similarity to calculate the mean of location. The median error distance of the RADAR system is about $2-3$ meters. Since the complexity of RADAR depends on the number of fingerprints in the database, RADAR suffers from long processing time when deployed in the large-scale area. Moreover, RADAR can suffer from the changing environment because it does not have the process to update the fingerprint database.

Recently, WinIPS [32] proposed the Gaussian Process Regression with Polynomial Surface Fitting Mean (PSFM-GPR) to predict received signal strength (RSS) values on each virtual reference point in order to construct a fingerprint. Then, for localization, the system leverages the Signal Tendency Index with Weighted K Nearest Neighbor (STI-WKNN) to estimate the user's location. Nevertheless, the processing time of WinIPS directly depends on the number of fingerprints in the database. So the algorithm takes a long processing time when deployed in the large-scale area. In addition, the algorithm wastes resources and time to process the queries which are sent from the out-of-scope area because it does not have any process to filter out those queries.

TABLE 2. Comparison of indoor localization systems based on Wi-Fi signal.

Indoor localization systems	Number of buildings tested per experiment	Multi-floors testing	Indoor/outdoor identification	Tested in large-scale area	Adapt to changing environment
RADAR [31]	1	×	×	×	×
WinIPS [32]	1	×	×	×	×
Horus [29]	1 (2 buildings were tested separately)	×	×	×	×
System proposed in [26]	1 (4 buildings were tested separately)	✓	✓	✓	×
Our new proposed system using *ExtHit*, *InHit* and *MissingHit*	37 (37 buildings were tested simultaneously)	✓	✓	✓	✓

We summarized the indoor localization systems based on Wi-Fi signal fingerprints in Table 2. Horus [29] was tested in two testbeds including one floor of the building in the university and one floor in office space. RADAR [31] was tested in one floor of the office space. WinIPS [32] was tested in one floor of the several layout spaces. However, Horus, RADAR and WinIPS have not been implemented, evaluated and discussed in multi-floor building, large-scale and indoor/outdoor scenarios. The system proposed by He *et al.* [26] was tested in a large-scale area including three floors in a business building, three floors in a shopping mall, two floors in an international airport and five floors in a university campus. The system can identify the location of the query whether it is indoor or outdoor.

Although Horus [29] and the system proposed by He *et al.* [26] were evaluated in many buildings, those buildings were tested separately in each experiment. Moreover, each building in the tested scenarios was far away from each other. Thus, these approaches are not effective in the large-scale exhibition where many buildings are closely located. Furthermore, these approaches are not adaptive to

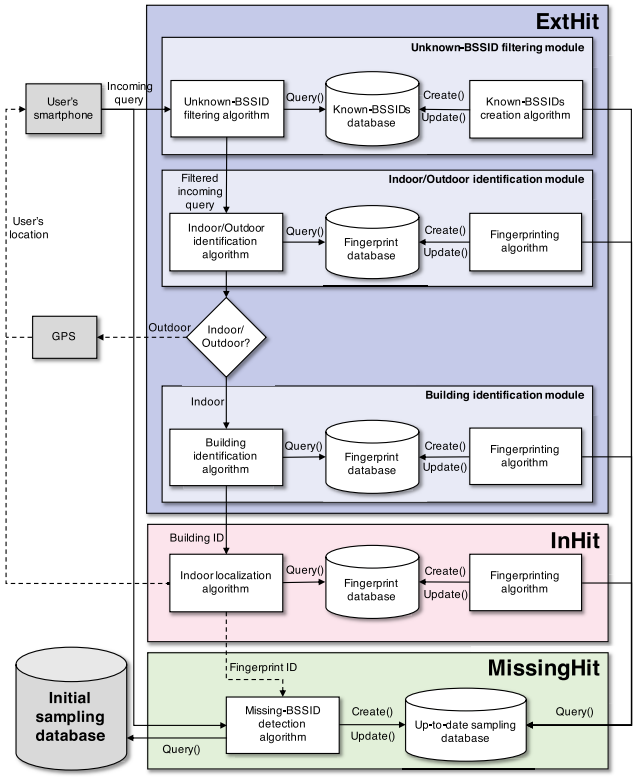


FIGURE 2. System architecture.

the changing environment which is the main problem that can degrade the performance of the system. Our proposed system using *ExtHit*, *InHit* and *MissingHit* is evaluated in a large-scale exhibition including 37 multi-floor buildings which are located close to each other. Our proposed *ExtHit* can identify the user's query whether it is sent from outdoor or a specific building. Then, it sends this information to our proposed *InHit*. *InHit* effectively localizes the exact location of the user based on the information from *ExtHit*. Furthermore, *MissingHit* utilizes those user's queries to detect the missing BSSIDs and adapts the fingerprint database to the changing environment. In aspects of processing time, according to the evaluation results in Section IV.C.8, *ExtHit* significantly outperforms the system in [26]. This is because the complexity of *ExtHit* is much lower and *ExtHit* does not require feature extraction.

III. ARCHITECTURE DESIGN

A. ARCHITECTURE OVERVIEW

In order to identify approximate location of users, we assume that each building or area has different characteristics of the Wi-Fi signal depending on its own environment.

Figure 2 illustrates our system architecture. The system consists of three main parts: First, our area classification algorithm, named *ExtHit*, identifies the area of the user which is outdoor or located in a specific building. If the algorithm can identify the building, the building ID will be sent to the next part. Otherwise, an outdoor location service, such as Global Positioning System (GPS), will perform. Second, our fingerprint-based indoor localization algorithm, named *InHit*, uses the building ID from the previous part to limit searching space in the fingerprint database. Then, the system returns the user's location as a result. Third, our missing-BSSID detector algorithm, named *MissingHit*, detects the missing BSSIDs in the incoming query and updates a sampling database.

ExtHit consists of three main modules: an unknown-BSSID filtering module, an indoor/outdoor identification module and a building identification module. The unknown-BSSID filtering module filters out the unknown BSSIDs in the incoming query. As a consequence, the filtered incoming query will be used as an input of the indoor/outdoor identification module. Next, the indoor/outdoor identification module identifies the filtered incoming query whether it is located indoor or outdoor. If the module returns an indoor as an output, it will proceed to the building identification module. The building identification module identifies which building the user is located. Finally, *ExtHit* sends the building ID to *InHit*, which is our indoor localization algorithm.

Our unknown-BSSID filtering module, indoor/outdoor identification module and building identification module are described in detail in Sections III.B III.C and III.D, respectively.

B. UNKNOWN-BSSID FILTERING MODULE

In practice, there are many unknown BSSIDs appearing due to the installation of Wi-Fi access points. Moreover, the queries from the area located outside the system's coverage area contain all of the unknown BSSIDs. These situations can degrade the performance of the overall system.

To mitigate the above-mentioned problem, we design the unknown-BSSID filtering module to remove unknown BSSIDs from the user's query. First, a known-BSSIDs creation algorithm creates a single list of known BSSIDs from all of the scanning results which are collected in the training phase and stored in the database. Then, an unknown-BSSIDs filtering algorithm filters out the unknown BSSIDs in the incoming query. Then, it returns the filtered incoming query which contains only the list of the known BSSIDs as a result.

The pseudocode for the known-BSSIDs creation algorithm and the unknown-BSSIDs filtering algorithm are shown in Algorithm 1 and 2, respectively. The complexity of Algorithm 1 is $O(sn)$ where s is the number of Wi-Fi scanning results obtained in the training phase and n is the maximum number of BSSIDs in the Wi-Fi scanning results. The complexity of the Algorithm 2 is $O(n)$ where n is the number of BSSIDs in the user's query.

C. INDOOR/OUTDOOR IDENTIFICATION MODULE

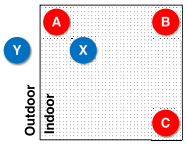
According to our experiments, we observed that Received Signal Strength Indicator (RSSI) values of Wi-Fi access points scanned at indoor positions are significantly higher than those scanned at outdoor positions. Figure 3 illustrates one of our experiments. Figure 3(a) shows the scanning positions and access points' locations. The filled-square is the building area which is indoor. The positions A, B and C are

Algorithm 1 Creating the List of Known

Input: collection of Wi-Fi scanning results *sc_list*
Output: known BSSIDs set *bssid_set*
1: Initialize *bssid_set*
2: **for each** *sc* **in** *sc_list* **do**:
3: **for each** *bssid* **in** *sc* **do**:
4: Add *bssid* to *bssid_set*
5: **end for**
6: **end for**
7: return *bssid_set*

Algorithm 2 Filtering the Unknown BSSIDs

Input: Wi-Fi scanning result from query *sc*,
known BSSIDs set from database *bssid_set*
Output: filtered Wi-Fi scanning result *filtered_sc*
1: Initialize *filtered_sc*
2: **for each** *bssid* **in** *sc* **do**:
3: **if** *bssid* is in *bssid_set* **then**
4: Add *bssid* to *filtered_sc*
5: **end if**
6: **end for**
7: return *filtered_sc*



(a)

Access point	RSSI
A	-43
B	-46
C	-55

(b)

Access point	RSSI
A	-69
B	-71
C	-85

(c)

FIGURE 3. Experimental results showing Wi-Fi scanning results at indoor and outdoor positions. (a) Scanning positions and access points' locations. (b) Wi-Fi scanning result at position X. (c) Wi-Fi scanning result at position Y.

Wi-Fi access points' locations. The positions X and Y are the scanning positions. Figure 3(b) and 3(c) show the Wi-Fi scanning results in order of signal strength at the positions X and Y, respectively. These results indicate that the orders of access points are the same at both positions. However, the RSSI values are significantly different. Therefore, we use this finding to design our indoor/outdoor identification module.

This indoor/outdoor identification module has two phases. First, a training phase collects Wi-Fi signals scanned at different positions in the indoor location in order to create the representative fingerprints for each building. Then, it calculates an *adaptive threshold score* which will be used as a parameter in the next phase. Second, a localization phase identifies the user's area which is located inside the building.

1) TRAINING PHASE

In the training phase, we collect Wi-Fi signals scanned at different positions in the indoor location, and create a representative fingerprint of each building. For each Wi-Fi scanning result, we sort and form it as

$$S_i = \left[s_i^1, s_i^2, s_i^3, \ldots, s_i^n\right] \quad (1)$$

Algorithm 3 Fingerprinting Algorithm for *ExtHit*

Input: top-N BSSIDs N,
collection of Wi-Fi scanning results *sc_list*,
building name *building_name*
Output: fingerprint *fingerprint*
1: Initialize *bssid_set*
2: **for each** *scanning_result* **in** *sc_list* **do**:
3: Initialize $count_N$ to 0
4: **for each** *bssid* **in** *scanning_result* **do**:
5: **if** $count_N$ is less than N **then**
6: Add 1 to $count_N$
7: **if** *bssid* **not in** *bssid_set*:
8: Add *bssid* to *bssid_set*
9: **end if**
10: **end if**
11: **end for**
12: **end for**
13: *fingerprint* ← {*building_name*, *bssid_set*}
14: return *fingerprint*

where S_i is the Wi-Fi scanning result i, s_i^j is the BSSID of access point j in S_i, n is the number of BSSIDs in S_i, and the RSSI of s_i^j is higher than or equal to the RSSI of s_i^{j+1}.

Our fingerprinting algorithm is used for creating the representative fingerprint which can represent the characteristics of indoor positions of each building. The algorithm uses the BSSIDs to identify the Wi-Fi access points. It combines N BSSIDs which provide the strongest signal strengths of each Wi-Fi scanning result in the building to create the representative fingerprint of the building. The pseudocode of our fingerprinting algorithm for *ExtHit* is shown in Algorithm 3. The top-N BSSIDs means the first N access points which provide the strongest signal strengths and are used to create the representative fingerprint of the building. The complexity of the algorithm is $O(Ns)$ when using top-N BSSIDs as an input of the algorithm and s is the number of Wi-Fi scanning results which are selected to create the representative fingerprint. After the creation, we have

$$FP = \{F_1, F_2, F_3, \ldots, F_b\} \quad (2)$$

where FP is a set of building's fingerprints, b is the number of buildings in the system and F_i is a fingerprint of building i which can be defined as

$$F_i = \left\{f_i^1, f_i^2, f_i^3, \ldots, f_i^m\right\} \quad (3)$$

where f_i^j is the BSSID of access point j in F_i, m is the number of BSSIDs in F_i

After the fingerprint creation, in order to calculate the *adaptive threshold score*, we transform the Wi-Fi scanning result by using 2 parameters including top-N BSSIDs, which are used in the fingerprint creation process, and the *pre-defined RSSI*. The BSSIDs which provide RSSI less than the *pre-defined RSSI* will be removed in the transformation

Algorithm 4 Indoor/Outdoor Identification Algorithm

Input: top-N BSSIDs N,
minimum RSSI *min_rssi*,
Wi-Fi scanning results from query *sc*,
list of fingerprints from database *fp_list*
Output: fingerprint score *score*
Initialize $count_N$ to 0
Initialize *score* to 0
for each *bssid* **in** *sc* **do**:
  if $count_N$ is less than N **then**
    Add 1 to $count_N$
    if *bssid.* rssi is more than or equal to *min_rssi*:
      for each *fp* **in** *fp_list* **do**:
        if *bssid* **in** *fp*:
          $score \leftarrow score + 1$
        end if
      end for
    end if
  end if
end for
return *score*

process. Thus, we have

$$T_i = \left[s_i^1, s_i^2, s_i^3, \ldots, s_i^l\right] \tag{4}$$

where T_i is a transformation result of S_i, s_i^k provides RSSI higher than or equal to the *pre-defined RSSI* and l is the number of BSSIDs in T_i. Then, the system calculates the score H_i of T_i using the Equation (5)

$$H_i = \max_{0 \le j \le b} \sum_{k=1}^{l} \begin{cases} 1, & s_i^k \in F_j \\ 0, & otherwise \end{cases} \tag{5}$$

The pseudocode of the score calculation is shown in Algorithm 4. The complexity of the algorithm is *O(bN)* when using top-N BSSIDs as an input of the algorithm and b is the number of the building's fingerprints. Finally, the system selects the minimum of the calculated scores to be the *adaptive threshold score* to use in the next phase.

2) LOCALIZATION PHASE

In this phase, whether the user's position is indoor or outdoor will be determined. Our indoor/outdoor identification algorithm is used. First, we form the incoming query as Equation (1). Next, we transform the incoming query by the methodology as shown in Equation (4). Then, we calculate the score of the incoming query using the Equation (5). Finally, if the score is higher than or equal to the *adaptive threshold score*, which means the scanning location of the incoming query is most likely to be inside the building, the algorithm will return *"indoor"* as a result. Otherwise, it will return *"outdoor"* as a result. This is because there is not enough matching between the BSSIDs in the building's fingerprint and those in the incoming query.

D. BUILDING IDENTIFICATION MODULE

In order to limit searching space, we design a low complexity algorithm to identify which building the user is located. A building is an appropriate scale to limit searching space for other algorithms because these algorithms consider the floor or area in the building.

We assume that each building has its own unique Wi-Fi signal fingerprint. The fingerprint can be represented by the BSSIDs which provide high signal strength when the scanning positions are in the building.

This building identification module consists of two phases. First, a training phase, the Wi-Fi scanning results are collected from target positions. The fingerprint of the building is created from those Wi-Fi scanning results in the building. It includes the Wi-Fi results scanned at the target positions on every floor of the building. Second, a localization phase identifies the building at which the user is located.

1) TRAINING PHASE

The process and the methodology of the fingerprinting algorithm are the same as the training phase of our indoor/outdoor identification module. The pseudocode of the fingerprinting algorithm is shown in Algorithm 3.

Note that the parameter N in the top-N BSSIDs for the building identification module can be different from that for the indoor/outdoor identification module. This is because each algorithm works separately and the parameter N which provides the best performance will be selected for each algorithm.

2) LOCALIZATION PHASE

In this phase, our building identification algorithm is used. The algorithm uses the concept of the nearest neighbor technique to identify the building by comparing the incoming query with the Wi-Fi signal fingerprints in the database. First, we form the incoming query as Equation (1) and transform the incoming query using the top-N BSSIDs which were used in the previous phase. Note that the top-N BSSIDs means the first N access points which provide the strongest signal strengths. Thus, we have,

$$Q = \left[q^1, q^2, q^3, \ldots, q^p\right] \tag{6}$$

where Q is a transformation result of the incoming query and p is the number of BSSIDs in Q. Then, the algorithm calculates a α value which represents the similarity between the incoming query and each building's fingerprint. The α value of building's fingerprint F_i can be calculated as Equation (7).

$$\alpha = \sum_{j=1}^{p} \begin{cases} 1, & q^j \in F_i \\ 0, & otherwise \end{cases} \tag{7}$$

After the calculation, the algorithm returns the building ID which provides the maximum α value as an answer. The pseudocode of the building identification algorithm is shown in Algorithm 5. The complexity of the algorithm is *O(bN)*

Algorithm 5 Building Identification Algorithm

Input: top-N BSSIDs N,
Wi-Fi scanning result from query *sc*,
list of fingerprints from database *fp_list*
Output: building ID *building_id*
1: Initialize $count_N$ to 0
2: **for each** *fp* **in** *fp_list* **do**:
3: Initialize *fp*. α to 0
4: **end for**
5: **for each** *bssid* **in** *sc* **do**:
6: **if** $count_N$ is less than N **then**
7: Add 1 to $count_N$
8: **for each** *fp* **in** *fp_list* **do**:
9: **if** *bssid* **in** *fp*:
10: Add 1 to *fp*. α
11: **end if**
12: **end for**
13: **end if**
14: **end for**
15: Initialize α to 0
16: **for each** *fp* **in** *fp_list* **do**:
17: **if** *fp*. α is more than α **then**
18: $\alpha \leftarrow$ *fp*. α
19: *building_id* $\leftarrow$ *fp*.building_id
20: **end if**
21: **end for**
22: return *building_id*

when using top-N BSSIDs as an input of the algorithm and b is the number of buildings' fingerprints.

E. InHit ALGORITHM

InHit is a fingerprint-based 3D indoor localization algorithm. It consists of two phases. First, the training phase collects Wi-Fi signals from target locations and creates a fingerprint map in the database. Second, the localization phase localizes the floor and the exact position on the floor at which the user is located.

1) TRAINING PHASE

The training phase for *InHit* algorithm is different from that for *ExtHit* algorithm. The fingerprinting algorithm for *InHit* will create the Wi-Fi signal fingerprints for every target location. For each Wi-Fi scanning result of each target location, we also form it as Equation (1). Then, the algorithm selects top-N BSSIDs which provide the strongest signal strength from the Wi-Fi scanning result to create a Wi-Fi signal fingerprint for the location. Thus, we have a set of Wi-Fi fingerprints, denoted as GP, defined as

$$GP = \{G_1, G_2, G_3, \ldots, G_r \tag{8}$$

where r is the number of fingerprints in the module's database, G_i is the fingerprint i which can be defined as

$$G_i = \left[s_i^1, s_i^2, s_i^3, \ldots, s_i^t\right] \tag{9}$$

Algorithm 6 Fingerprinting Algorithm for *InHit*

Input: top-N BSSIDs N,
collection of Wi-Fi scanning results *sc_list*,
floor level *floor_level*
position *(x,y)*
Output: fingerprint *fingerprint*
1: Initialize $count_N$ to 0
2: Initialize *bssid_list*
3: **for each** *bssid* **in** *sc_list* **do**:
4: **if** $count_N$ is less than N **then**
5: Add 1 to $count_N$
6: Add *bssid* to *bssid_list*
7: **end if**
8: **end for**
9: *fingerprint* $\leftarrow$ {*floor_level, (x, y), bssid_set*}
10: return *fingerprint*

where t is the number of BSSIDs in G_i. The pseudocode of the fingerprinting algorithm is shown in Algorithm 6. The complexity of the algorithm is $O(N)$ when using top-N BSSIDs as an input. Note that the parameter N in the top-N BSSIDs for the *InHit* algorithm can be different from that for the *ExtHit* algorithm. This is because each algorithm works separately and the parameter N which provides the best performance will be selected for each algorithm.

2) LOCALIZATION PHASE

We also transform the incoming query as Equation (6) by using the same parameter as the training phase of this module. Then, the algorithm calculates a β value which represents the similarity between the transformation result of the incoming query and each fingerprint in the database. Then, the algorithm returns the floor and position on the floor of the fingerprint that provides the maximum β value.

The system calculates the β value of the incoming query, denoted as Q, and fingerprint G_i using Equation (10)

$$\beta = \sum_{j=1}^{p} \sum_{k=1}^{t} \begin{cases} 1, & q^j = s_i^k \text{ and } \left|RSSI\left(q^j\right) - RSSI\left(s_i^k\right)\right| \leq \gamma \\ 0, & otherwise \end{cases} \tag{10}$$

where $RSSI$ (s) is RSSI of s and γ is a predefined value for the algorithm.

The pseudocode of the localization algorithm is shown in Algorithm 7. The complexity of the algorithm is $O(rN^2)$ when using top-N BSSIDs as an input and r is the number of fingerprints in the database.

F. MissingHit ALGORITHM

In a real situation, many BSSIDs which we surveyed can be missed due to the access points replacement or the closed offices. These situations degraded the performance of the overall system. To tackle the problems, we propose a missing-BSSID detector module, *MissingHit*, to detect the missing BSSIDs from the user's query and remove those

Algorithm 7 Localization Algorithm for *InHit*

Input: Wi-Fi scanning result *sc*,
pre-defined value γ,
fingerprints from the database *fp_list*,
Output: floor level *floor_level*,
position *position*

```
for each fp in fp_list do:
    Initialize fp. β to 0
    for each bssid_sc in sc do:
        for each bssid_fp in fp do:
            if bssid_sc equals to bssid_fp:
                if |RSSI(bssid_sc)–RSSI(bssid_fp)| ≤ γ
                    Add 1 to fp.β
                end if
            end if
        end for
    end for
end for

Initialize β to 0
for each fp in fp_list do:
    if fp. β is more than β then
        β ← fp. β
        floor_level ← fp.floor_level
        position ← fp.position
    end if
end for
return floor_level, position
```

BSSIDs in the sampling database to maintain the performance of the overall system.

When user queries to the system for localization, the system duplicates the incoming query to this module and we form the incoming query, defined as R_i, using Equation (1). Then, when the system completes the localization process, the system also sends the fingerprint ID of the most matched fingerprint, using the *InHit* algorithm, to this module in order to detect the missing BSSIDs.

For the missing-BSSIDs detection process, this process will work once a day because, in our case, we deploy the system in the exhibition. Thus, the environment can be changed everyday. Thus, if the system is deployed in different conditions, the working period of this process can be changed. Note that, in order to set the working period, using a few user's queries to detect the missing BSSIDs is not practical because some BSSIDs may temporarily disappear from the scanning result and should not be deleted. Therefore, the system should wait for a reasonable period of time in order to detect the missing BSSIDs.

When it is time to update the sampling database, such as the end of the day, a missing-BSSID detection algorithm will be called. For each pair of incoming query and its fingerprint ID, the algorithm queries an *initial sampling database* for the Wi-Fi scanning result which matches the fingerprint ID and we also form the Wi-Fi scanning result, defined as P_i, using Equation (1). Then, the algorithm creates a *possible missing-BSSIDs set* of R_i, defined as MS_i, using the Equation (11)

$$MS_i = \{x | x \in P_i, x \notin R_i\} \tag{11}$$

and the algorithm creates a *found-BSSIDs set* of R_i, defined as FS_i, using the Equation (12)

$$FS_i = \{x | x \in R_i\} \tag{12}$$

Next, the algorithm creates the *actual missing-BSSIDs set*, defined as *AMS* by using Equation (13)

$$AMS = \cup_{i=1}^{u} MS_i - \cup_{i=1}^{u} FS_i \tag{13}$$

where u is the number of incoming queries which are used for updating the sampling database. Finally, the module updates the sampling database by duplicating the *initial sampling database* to *up-to-date sampling database* and removing the BSSIDs which exist in the *AMS* set. Therefore, when the other modules update their fingerprint database, the fingerprint databases will not include those missing BSSIDs which are detected by this module. The pseudocode for creating the *AMS* is shown in Algorithm 8. The complexity of the algorithm is $O(un)$ where n is the number of BSSIDs in the incoming query and u is the number of incoming queries which are used for updating the sampling database.

Algorithm 8 Creating the Set of Actual Missing-BSSIDs

Input: list of missing-BSSIDs set *MS*,
list of found-BSSIDs set *FS*,
Output: set of actual missing-BSSIDs *AMS*

```
Initialize AMS to { }
Initialize U_MS to { }
for each ms_i in MS do:
    U_MS ← U_MS unionms_i
end for

Initialize U_FS to { }
for each fs_i in FS do:
    U_FS ← U_FS unionfs_i
end for

AMS ← U_MS – U_FS
return AMS
```

IV. PERFORMANCE EVALUATION

A. EXPERIMENTAL SETUP

1) EXTHIT

We evaluated the accuracy of *ExtHit* in the large-scale campus exhibition named Chula Expo 2017. The data for accuracy evaluation were collected in 37 multi-floor buildings in Chulalongkorn University. The map of the exhibition area in Chulalongkorn University is shown in Figure 1. We gathered 5 datasets in which the 1st, the 2nd and the 3rd datasets were collected before the exhibition, the 4th dataset was collected

TABLE 3. The number of collected Wi-Fi scanning results and details for each dataset.

Data-set	Date collected	Environment		Number of Wi-Fi scanning results	Number of buildings
		Indoor	Outdoor		
1	2 weeks before exhibition days	✓	✓	1010	2
2	2 weeks before exhibition days	✓	×	2,239	37
3	2 weeks before exhibition days	✓	×	2,260	37
4	Exhibition days	✓	×	1,142	34
5	Exhibition days	Unknown	Unknown	609,277	Unknown

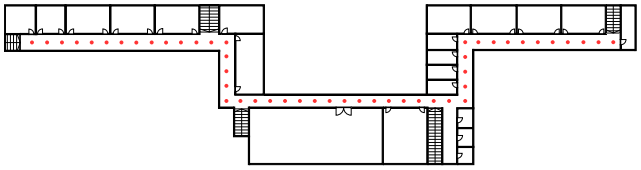


FIGURE 4. The floor plan of the ENG3 building.

in the exhibition days, and the 5th dataset was collected from the real users' queries in the exhibition days. We gathered data by continuously sampling along the walkway in the building for the indoor data, and sampling around the walkway outside the building within $2-3$ meters from the building's wall for the outdoor data. The details for each dataset are shown in Table 3. We use one Samsung Galaxy S5 and two LG Nexus 5X to collect data. The two LG Nexus 5X are defined as *Nexus 5X* (*A*) and *Nexus 5X* (*B*) to differentiate between these devices.

2) INHIT

InHit was initially evaluated in the ENG3 building of the Faculty of Engineering, Chulalongkorn University. We collected 3,175 Wi-Fi scanning results from 127 different positions on the 1st, the 2nd and the 3rd floors of the building. We divided the area along the walkway into a grid with the equal size. The length of each grid is 4 meters and the center of each grid is the coordinate of the reference point to collect the Wi-Fi scanning results. The floor plan of the building is shown in Figure 4. The red dots in the figure are the scanning positions (says, the reference points of each grid).

We use one Samsung Galaxy Note 4, one Samsung Galaxy S5 and one LG Nexus 5X to collect data.

3) SIMULATION

In order to evaluate the impact of the changing environment including missing BSSIDs and varying RSSI, we created the simulation of these scenarios.

In the case of missing BSSIDs, we randomly select the BSSIDs from the testing data with the number of BSSIDs which we require to remove. Then, we remove the selected BSSIDs in the testing data before we begin the evaluation in order to evaluate the impact of the missing BSSIDs.

In the case of varying RSSI, we also randomly select the BSSIDs from the testing data with the number of BSSIDs which we require to vary the RSSI. Next, we vary the RSSI of selected BSSIDs of each testing data using the Equation (14)

$$RSSI_{new} = 10 \times log_{10}(10^{\frac{RSSI_{old}}{10}} \times \omega) \tag{14}$$

where $RSSI_{new}$ is the varied RSSI value, $RSSI_{old}$ is the original RSSI value and ω is the percent of remaining transmission power of the access point which provides the BSSID.

Moreover, to mitigate the varying result due to the randomization, we repeat the simulation for 10 times with different random seeds and average the results of these simulations.

B. METRIC

1) ACCURACY

Accuracy is measured as a percentage of the number of correct results using Equation (15). In the case of the indoor/outdoor identification module, $n_{correct}$ is the number of correct area results. In the case of the building identification module, $n_{correct}$ is the number of correct building results. In the case of the indoor localization algorithm, $n_{correct}$ is the number of correct floor results and n_{all} is the number of tested data.

$$Accuracy = \frac{n_{correct}}{n_{all}} \times 100 \tag{15}$$

2) ERROR DISTANCE

Error distance is measured as a distance between the real scanning position and the localized position using Euclidean distance as in Equation (16) where $\mathrm{x} = \{x_1, x_2\}$ is the real scanning position and $\mathrm{y} = \{y_1, y_2\}$ is the localized position.

$$Error\ distance = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \tag{16}$$

3) CUMULATIVE PROCESSING TIME

Cumulative processing time is measured as a summation of processing time of the algorithm on a set of data using Equation (17) where s_i is the ith sampling, $p(s_i)$ is the processing time of the algorithm on the ith sampling and n is the number of tested samplings.

$$Cumulative\ processing\ time = \sum_{i=1}^{n} p(s_i) \tag{17}$$

C. EXPERIMENTAL RESULTS

1) INDOOR/OUTDOOR IDENTIFICATION MODULE

In order to determine the appropriate values of top-N and the *pre-defined RSSI*, we varied the number of top-N BSSIDs

TABLE 4. The maximum accuracy of the indoor/outdoor identification module for each parameter.

Tested with	Accuracy (%)	*Pre-defined RSSI*	top-*N*
Nexus 5X	93.27	-88	77
Samsung S5	89.11	-77	30

and the *pre-defined RSSI* to investigate the accuracy. Moreover, we also tested with various models of mobile devices. We used the 1st dataset to evaluate the module because this dataset consists of the samplings collected at both indoor and outdoor locations.

We started the test with top-1 BSSID and −90 dBm as the *pre-defined RSSI*. Then, we increased the number of top-*N* and the *pre-defined RSSI* to top-80 and −55 dBm, respectively. The accuracy results of the indoor/outdoor identification module are shown in Figure 5. The maximum accuracy for each case is summarized in Table 4.

According to Table 4, the highest accuracy can be achieved when tested with Nexus 5X. This might be because the wireless interface of LG Nexus 5X is a newer technology than Samsung Galaxy S5. Thus, LG Nexus 5X is more sensitive to signal than Samsung Galaxy S5. As can be seen in overall, the maximum accuracy that the module achieves in every case of mobile devices are slightly different. Thus, the indoor/outdoor identification module can operate for various devices.

In order to evaluate the impact of the changing environment including missing BSSIDs and varying RSSI, we simulated those scenarios with the simulation as we mentioned in Section IV.A.3. by using the 1st dataset.

In order to select the optimal parameter for the simulation, as can be seen in the Figure 5, the algorithm achieves high accuracy for both devices when utilizes −86 to −90 dBm as the *pre-defined RSSI* and top-30 to top-40 BSSIDs as an input of the algorithm. Thus, we selected the average values which are −88 dBm as the *pre-defined RSSI* and top-35 BSSIDs as an input of the algorithm for this simulation.

In the case of missing BSSIDs, we started the test by removing 10% of the BSSIDs from the testing data. Then, we increased the number of removed BSSIDs to 70%. The accuracy results of the simulation are shown in Table 5. As can be seen in the table, the accuracy significantly decreases when the number of missing BSSIDs are more than 10%. This is because, when more than 10% of the BSSIDs are removed, the calculated score according to the Equation (5) decreases and becomes less than the *adaptive threshold score*. As a result, the algorithm determines it as the incoming query that is sent from outdoor although it is sent from indoor. This can degrade the overall performance of the system. In Section IV.C.5, we will show that using our *MissingHit* can tackle the problem and therefore improve the accuracy in these scenarios.

In the case of varying RSSI, we started the test by selecting 25% of the BSSIDs from the testing data and reducing the

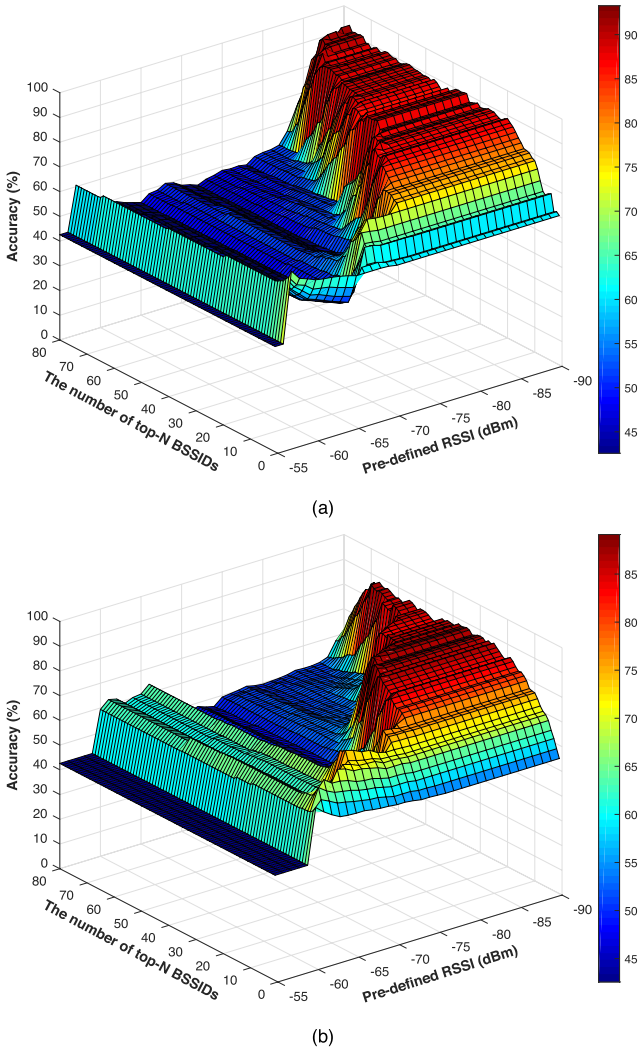


FIGURE 5. Accuracy of the indoor/outdoor identification module. (a) Tested with LG Nexus 5X. (b) Tested with Samsung Galaxy S5.

TABLE 5. Accuracy of the indoor/outdoor identification module missing-BSSID scenarios.

Removed BSSIDs	0%	10%	20%	30%	40%	50%	60%	70%
Accuracy (%)	90.30	88.67	84.38	80.02	74.50	70.20	64.89	58.79

transmission power of the selected BSSIDs to 90% of the original transmission power. Then, we increased the number of selected BSSIDs to 75% and reduced the transmission power to 40%. The accuracy results of these scenarios are shown in Table 6. As can be seen in overall, the accuracy for each case is slightly different. This is because, according to the Equation (14) which describes the relation between RSSI value and transmission power, the RSSI value has logarithmic relation with the transmission power. Thus, if the number of top-*N* applied is high enough, the algorithm can still tolerate this changing RSSI value. Therefore, the indoor/outdoor identification module is robust to varying RSSI.

TABLE 6. Accuracy of the indoor/outdoor identification module in varying-RSSI scenarios.

Selected BSSIDs	Remaining power						
	100%	90%	80%	70%	60%	50%	40%
25%	90.30	90.30	90.30	90.32	90.77	90.99	90.99
50%	90.30	90.30	90.30	90.57	91.12	90.43	90.44
75%	90.30	90.30	90.30	90.79	91.01	90.25	90.26

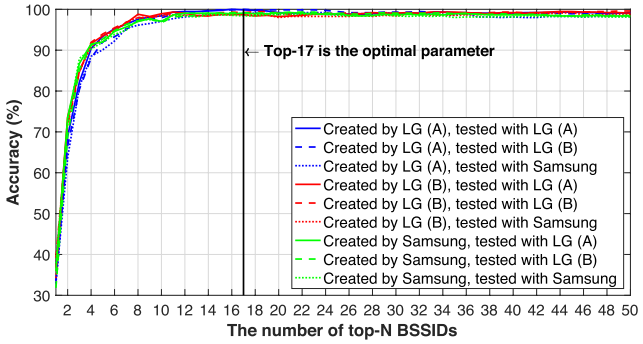


FIGURE 6. Accuracy of the building identification module.

2) BUILDING IDENTIFICATION MODULE

We used the 2nd and the 3rd as datasets for evaluating the building identification module. This is because these datasets were collected in many buildings inside the exhibition area.

In order to evaluate the impact of mobile devices, we tested the module with various numbers of top-*N* BSSIDs and varied models of mobile devices.

Figure 6 shows the accuracy results of the building identification module. We began the test with top-1 BSSIDs and we increased the number of top-*N* BSSIDs to top-50 for every case of mobile devices. Table 7 summarizes the maximum accuracy for each case.

According to Table 7, when tested with LG Nexus 5X the module provides higher maximum accuracy than when tested with Samsung Galaxy S5 in every case. This might be because the IEEE 802.11 wireless chip of LG Nexus 5X is a newer technology than Samsung Galaxy S5. Therefore, LG Nexus 5X is more sensitive to Wi-Fi signal than Samsung Galaxy S5.

As can be seen in overall, the maximum accuracy for each case is slightly different. Thus, the building identification module is robust to heterogeneous devices.

In order to select the optimal parameters for the building identification module, we considered the above-mentioned results and concluded that *LG Nexus 5X* (*A*) and top-17 is the most appropriate condition. This is because it can provide 100% accuracy when tested with the same device model and 98.93% accuracy when tested with other device models. Nevertheless, according to Figure 6, the accuracy tends to be stable when the algorithm utilized more than top-10. Therefore, if one concerns about the processing time and can accept a little lower accuracy, we suggest top-11 as an input of the algorithm to reduce the overall processing time.

TABLE 7. The maximum accuracy of the building identification module for each parameter.

Creating fingerprint by	Tested with	Accuracy (%)	top-*N*
Nexus 5X (A)	Nexus 5X (A)	100.00	16
	Nexus 5X (B)	100.00	17
	Samsung S5	99.20	22
Nexus 5X (B)	Nexus 5X (A)	99.46	44
	Nexus 5X (B)	99.59	49
	Samsung S5	99.20	45
Samsung S5	Nexus 5X (A)	99.19	21
	Nexus 5X (B)	99.31	21
	Samsung S5	98.80	21

TABLE 8. Accuracy of the building identification module in missing-BSSIDs scenarios.

Removed BSSIDs	0%	10%	20%	30%	40%	50%	60%	70%
Accuracy (%)	100	97.39	97.14	97.02	96.88	96.53	96.1	95.22

TABLE 9. Accuracy of the building identification module in varying-RSSI scenarios.

Selected BSSIDs	Remaining power						
	100%	90%	80%	70%	60%	50%	40%
25%	100	99.73	99.73	99.65	99.61	99.62	99.62
50%	100	99.73	99.73	99.64	99.61	99.62	99.62
75%	100	99.73	99.73	99.64	99.61	99.64	99.64

Moreover, we also evaluated the module in scenarios where there exist missing BSSIDs and varying RSSI by using *LG Nexus 5X* (*A*) as a tested device. In the case of missing BSSIDs, we began the test by removing 10% of the BSSIDs and increased to 70% of the BSSIDs. We used top-17 as an input of the algorithm because it provides the highest accuracy when tested with all device models. The simulation results are shown in Table 8. As can be seen in the table, the accuracy slightly decreases when the number of removed BSSIDs increases. In Section IV.C.5, we will show that using our *MissingHit* can improve the accuracy in these scenarios.

In the case of varying RSSI, we began the test by selecting 25% of the BSSIDs from the testing data and reducing the transmission power of the selected BSSIDs to 90% of the original transmission power. Then, we increased the number of selected BSSIDs to 75% and reduced the transmission power to 40%. The accuracy results of these scenarios are shown in Table 9. As can be seen in the table, the accuracy for each case is slightly different. Thus, the building identification module is robust to the scenarios where there exist varying RSSI.

3) BUILDING IDENTIFICATION MODULE EVALUATED IN THE REAL CHANGING ENVIRONMENT

We summarize the number of the unique BSSIDs before the exhibition days and during the exhibition in Figure 7 using the 2nd and the 4th datasets. The number of BSSIDs which were found in both before the exhibition days and during the

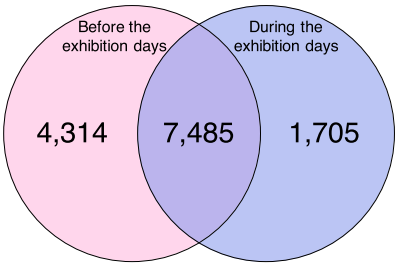


FIGURE 7. Changing of the number of the unique BSSIDs between before the exhibition days and during the exhibition days.

exhibition days is 7,485 BSSIDs. The number of BSSIDs which were found only before the exhibition (say, missed BSSIDs) is 4,314 BSSIDs which is 36% of the BSSIDs that were found before the exhibition days. The number of BSSIDs which were found only during the exhibition days (say, unknown BSSIDs) is 1,705 BSSIDs. As can be seen, the number of the unique BSSIDs are critically changed.

In order to evaluate the impact of the changing environment. We use the 2^{nd} and the 4^{th} datasets to evaluate the building identification module. This is because the 2^{nd} dataset was collected before the exhibition days and the 4^{th} dataset was collected during the exhibition days.

We tested the module with various numbers of top-N BSSIDs and varied models of mobile devices.

Figure 8 shows the accuracy results of the module. We began the test with top-1 BSSIDs and we increased the number of top-N to top-50 for every case of mobile devices. Table 10 summarizes the maximum accuracy for each case.

According to Table 10, the overall accuracy in this evaluation is less than that in the previous evaluation. Applying the higher top-N tends to achieve the higher accuracy. This is because of the impact of the changing environment where the number of the unique BSSIDs are significantly different. However, the maximum accuracy for each case slightly reduces (approximately $3-7\%$). Thus, it can be concluded that the building identification module is robust to the changing environment. Moreover, these results are related to our simulation results in Section IV.C.2 in that the accuracy results decrease approximately 3% when removing 30∼40% of BSSIDs and using Nexus 5X.

4) INHIT

We used the dataset collected in the ENG3 building as described in Section IV.A.2 to evaluate our *InHit* indoor localization algorithm. We tested the algorithm with various number of top-N BSSIDs and γ which is the predefined value. Moreover, we also used various types of mobile devices to evaluate the impact of these factors.

We used the dataset collected by Samsung Galaxy Note 4 to be a training dataset. Next, we began the test with top-20 BSSID and 0 dBm as the γ value. Then, we increased the number of top-N BSSIDs and the γ value to top-200 and

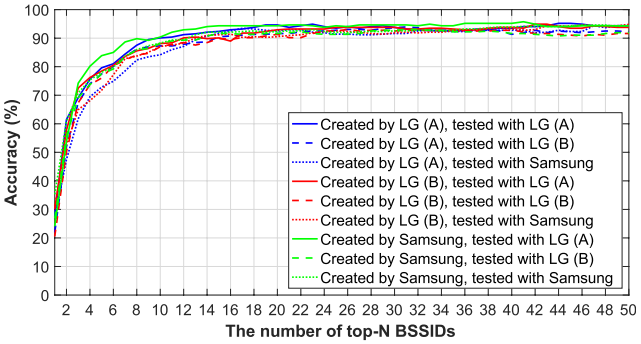


FIGURE 8. Accuracy of the building identification module evaluated in the changing environment.

TABLE 10. The maximum accuracy of the building identification module evaluated in the changing environment.

Creating fingerprint by	Tested with	Accuracy (%)	top-N
Nexus 5X (A)	Nexus 5X (A)	95.18	44
	Nexus 5X (B)	93.93	31
	Samsung S5	94.34	47
Nexus 5X (B)	Nexus 5X (A)	94.90	42
	Nexus 5X (B)	93.06	26
	Samsung S5	94.80	50
Samsung S5	Nexus 5X (A)	95.75	41
	Nexus 5X (B)	93.06	31
	Samsung S5	94.57	46

TABLE 11. The maximum accuracy for floor localization and the minimum average error distance for localization of InHit algorithm.

Tested with	Floor accuracy (%)	Parameter (floor)	Average error distance (m.)	Parameter (error distance)
Samsung Note 4	100	top-80, $\gamma = 2$	0.12	top-180, $\gamma = 4$
Samsung S5	100	top-180, $\gamma = 5$	0.47	top-180, $\gamma = 7$
Nexus 5X	100	top-120, $\gamma = 3$	0.63	top-180, $\gamma = 8$

10 dBm, respectively. Figure 9 and figure 10 show the accuracy of the floor localization and the average error distance in the *InHit*, respectively. Table 11 summarizes the maximum accuracy for floor localization and the minimum average error distance for indoor localization.

As can be seen in figure 9, the accuracy for floor localization of *InHit* tends to utilize higher top-N to achieve 100% of accuracy when tested with other mobile devices. In figure 10, the average error distance of *InHit* tends to critically decrease from top-20 to top-80 and tends to slightly decrease when the algorithm uses more than top-80. In addition, the algorithm requires higher γ value to achieve lower average error distance when tested with other device models. This is because the different device models are embedded with the different wireless interfaces. Therefore, the RSSI values provided by those wireless interfaces are varied. In overall, *InHit* achieves up to 100% of accuracy for floor localization. Its average error distance for 3D localization is approximately 0.12 to 0.63 meters.

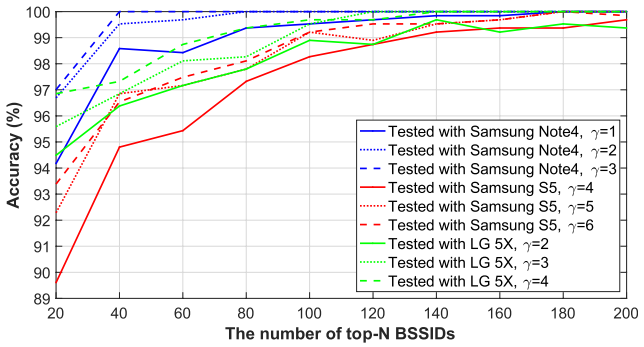


FIGURE 9. Accuracy of the floor localization in InHit algorithm.

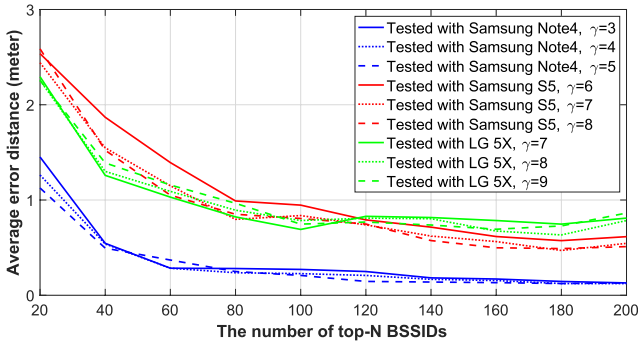


FIGURE 10. Average error distance of the 3D localization in InHit algorithm.

Furthermore, we evaluated *InHit* in the changing environment scenarios including missing BSSIDs and varying RSSI. We used the dataset collected by Samsung Galaxy Note 4 for the simulation as described in Section IV.A.3. In the simulation, we used top-180 and 4 dBm as the γ value for the input of the algorithm because the algorithm achieves the maximum floor localization accuracy and the minimum average error distance when utilizes these values.

In the case of missing BSSIDs, we started the test by removing 10% of the BSSIDs from the testing data. Then, we increased to 70% of the BSSIDs. Table 12 summarized the average error distance results of the impact of missing BSSIDs. According to the table, the average error distance of each case slightly decreases when the number of removed BSSIDs are increased. This is because, when we use a high number of top-N BSSIDs as an input of the algorithm, the β value is not affected much by the number of removed BSSIDs. So the algorithm can still tolerate this change. Thus, *InHit* is robust to the scenarios where there exist missing BSSIDs.

In the case of varying RSSI, we started the test by selecting 25% of the BSSIDs from the testing data and reducing the transmission power of the selected BSSIDs to 90% of the original transmission power. Then, we increased the number of selected BSSIDs to 75% and reduced the transmission power to 40%. Table 13 summarized the average error distance results of the impact of varying BSSIDs. As can be seen

TABLE 12. The average error distance results of InHit algorithm in missing-BSSIDs scenarios when using top-180 and 4 dBm as the γ value for the input.

Removed BSSIDs	10%	20%	30%	40%	50%	60%	70%
Average error distance (meter)	0.15	0.18	0.2	0.19	0.2	0.21	0.26

TABLE 13. The average error distance results of InHit algorithm in varying-RSSI scenarios when using top-180 and 4 dBm as the γ value for the input.

Selected BSSIDs	Remaining power					
	90%	80%	70%	60%	50%	40%
25%	0.12	0.12	0.13	0.14	0.14	0.14
50%	0.12	0.12	0.14	0.16	0.17	0.17
75%	0.12	0.12	0.15	0.18	0.2	0.2

TABLE 14. The missing-BSSID detection rate results of MissingHit.

Removed BSSIDs	10%	20%	30%	40%	50%	60%	70%
Detection rate (%)	97.55	97.81	97.94	97.72	98.03	97.90	97.76

in the table, the average error distance is slightly different in each case and the analysis of this simulation results are already described in Section IV.C.1. Thus, *InHit* is robust to the scenarios where there exists varying RSSI.

According to the simulation results, *InHit* is robust to the changing environment including missing BSSIDs and varying RSSI.

5) MISSINGHIT

We used the simulation as described in Section IV.A.3 and used the dataset as described in Section IV.A.2 to evaluate our missing-BSSID detector module, *MissingHit*. We started the test by removing 10% of the BSSIDs and increased to 70% of the BSSIDs. According to the results of the *InHit* algorithm in Section IV.C.4, the algorithm achieves high accuracy for localization for both normal scenario and missing-BSSIDs scenarios when utilizing top-180 and 4 dBm as the γ value. Thus, we used those parameters as an input of the algorithm to evaluate the module. The missing-BSSID detection rate results are summarized in Table 14. According to the table, the module can achieve a high missing-BSSID detection rate in every case (approximately 97.55–98.03%). Therefore, this module can be used to detect the missing-BSSID efficiently. Then, the system can update the sampling database to tackle the changing environment problems.

Since our indoor/outdoor identification module is not robust to missing BSSIDs, we will use the *MissingHit* to update the sampling database before the indoor/outdoor identification module creates its fingerprint database. We used the 1st dataset, as described in Section IV.A.1, which collected by *Nexus 5X* (*A*) and *Nexus 5X* (*B*) as the training andv testing data in this evaluation.

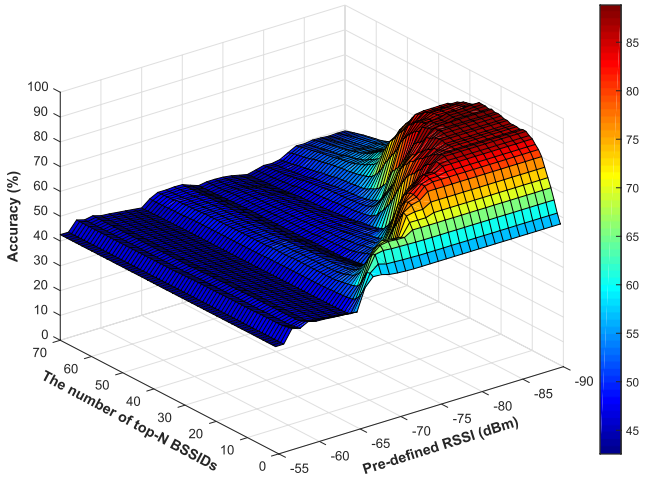


FIGURE 11. Accuracy of the indoor/outdoor identification module with MissingHit at 50% of BSSIDs are missed.

Figure 11 shows the accuracy results of the indoor/outdoor identification module with the *MissingHit* in the scenario that there are 50% of missing BSSIDs. The maximum accuracy of the indoor/outdoor identification module is 88.83% when using top-27 and −89 dBm as the *pre-defined RSSI*. As can be seen in the figure, the maximum accuracy tends to utilize lower top-*N* value compared to the normal scenario, which is shown in Figure 5. So we can select an optimal parameter for each situation in advance by observing from the simulation results. Thus, in practice, the indoor/outdoor identification module can adapt the parameter according to the number of missing BSSIDs in order to achieve the maximum accuracy for each situation.

Figure 12 shows the accuracy comparison of the indoor/outdoor identification module *without MissingHit* and the indoor/outdoor identification module *with MissingHit*. As can be seen in the figure, in the case of removing 70% of BSSIDs, the accuracy increases from 58.79% to 85.92%. Thus, the *MissingHit* can significantly improve the accuracy of the indoor/outdoor identification module.

Then, in order to improve the accuracy for the building identification module, we use the *MissingHit* to update the sampling database before the building identification module creates its fingerprint database. We used the 2nd and the 3rd datasets which were collected by *Nexus 5X* (*A*) and *Nexus 5X* (*B*) for this evaluation.

Table 15 shows the accuracy comparison of the building identification module *without MissingHit* and the building identification module *with MissingHit*. As shown in the table, the *MissingHit* can improve the accuracy of the building identification module in every scenario.

In overall, the algorithms have higher robustness to the changing environment when integrated with *MissingHit*. This is because *MissingHit* updates the sampling database before the indoor/outdoor identification module and the building identification module creates their fingerprint database.

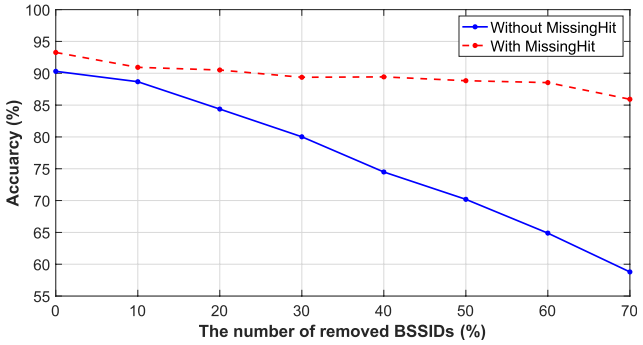


FIGURE 12. The accuracy comparison between the indoor/outdoor identification module without MissingHit and the indoor/outdoor identification module with MissingHit.

TABLE 15. The accuracy comparison between the building identification module without MissingHit and the building identification module with MissingHit.

Removed BSSIDs	10%	20%	30%	40%	50%	60%	70%
Without *MissingHit*	97.39	97.14	97.02	96.88	96.53	96.10	95.22
With *MissingHit*	99.85	99.80	99.67	99.43	99.07	98.74	98.27

6) PERFORMANCE COMPARISON WITH THE EXISTING AREA CLASSIFICATION TECHNIQUE

We compare the performance of our proposed *ExtHit* with that of the area classification system proposed in [26] because the system in [26] is recently proposed. Moreover, it achieves high accuracy and uses the Wi-Fi signal as an input. The system in [26] consists of three main modules. First, the one-class inside/outside-region detection module identifies the user's query whether it is located indoor or outdoor. Second, the area classification module determines building of the user. Third, the device calibration module calibrates the device's scanning results. We conducted two experiments as follows.

1st Experiment: the objective of this experiment is to evaluate the one-class inside/outside-region detection module of the system proposed in [26], and to compare with our indoor/outdoor identification module. The 1st dataset as shown in Table 3 is used in this experiment. Since Principal Component Analysis (PCA) is reported in [26] that it provides the best performance, it is applied in the one-class inside/outside-region detection module.

Moreover, we evaluated the module in the missing-BSSIDs scenarios by using the simulation as described in Section IV.A.3. Since our *MissingHit* can improve the accuracy in the changing environment, we also integrate the module with one-class inside/outside-region detection module of the system proposed in [26] and compare the accuracy results in Figure 13. In the normal scenario where there is no missing BSSIDs, the system proposed in [26] achieves 83.17% of accuracy while our proposed system achieves 93.27%. Furthermore, the accuracy of the system proposed in [26] significantly decreases in the missing-BSSIDs scenarios. This is because the missing-BSSIDs affect the *reconstruction*

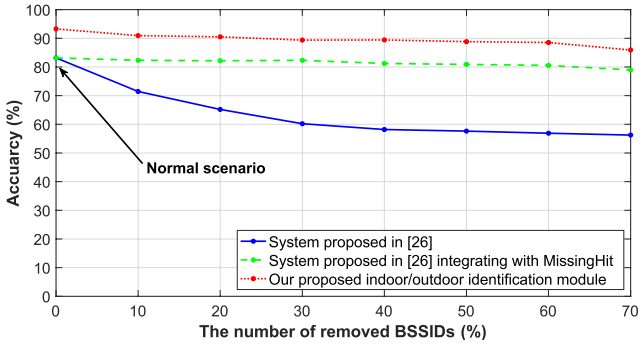


FIGURE 13. Accuracy of the one-class inside/outside-region detection module of the system proposed in [26] in the missing-BSSIDs scenarios.

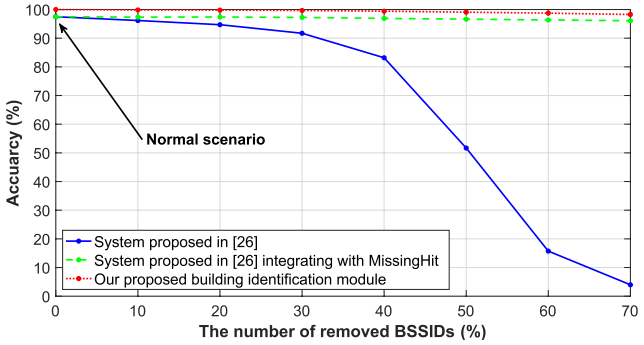


FIGURE 14. Accuracy of the area classification module of the system proposed in [26] compared to our proposed building identification module in the missing-BSSIDs scenarios.

error value which PCA utilizes to determine a region of incoming query. However, after integrating with our *MissingHit*, the accuracy of the system significantly increases, approximately 23.64%.

2nd Experiment: the objective of this experiment is to evaluate the area classification module of the system proposed in [26], and to compare with our building identification module. The 2nd and the 3rd datasets shown in Table 3 are used as a training data and a testing data, respectively. Since Probabilistic Support Vector Machine (SVM) is reported in [26] that it provides the best performance, it is applied in the area classification module.

Furthermore, we also used the simulation as described in Section IV.A.3 to evaluate the module in missing-BSSIDs scenarios. Then, we integrated our *MissingHit* with the area classification module of the system proposed in [26] to improve the accuracy in the changing environment. Figure 14 shows the accuracy results of this experiment. In the normal scenario where the number of removed BSSIDs is zero, the system proposed in [26] achieves 97.49% while our proposed system achieves 100%. In the missing-BSSIDs scenarios, the accuracy of the system proposed in [26] critically decreases when the number of removed BSSIDs is higher than 40%. When the number of removed BSSIDs is 70%, the accuracy of the system becomes 3.99% which is extremely low. This is because the more the missing-BSSIDs, the sparser the vector. This leads to inaccurate classification of SVM. However, when we help the system by integrating the system with our *MissingHit*, the accuracy significantly increases from 3.99% to 96.08%. This shows that our *MissingHit* is very helpful in the changing environment.

As can be seen in the both experiments, our proposed *ExtHit* can significantly outperform the system proposed in [26]. Moreover, our *MissingHit* can significantly improve the accuracy of the system proposed in [26] in the changing environment.

7) PERFORMANCE COMPARISON WITH THE EXISTING INDOOR LOCALIZATION TECHNIQUE

We compare *InHit* with the existing indoor localization systems named RADAR [31] and WinIPS [32]. The dataset described in Section IV.A.2 is used for this evaluation. We also evaluated those systems in the missing-BSSIDs scenarios by using the simulation as described in Section IV.A.3. Moreover, we also integrate our *MissingHit* to improve the accuracy for those systems. Table 16 and Figure 15 show the floor localization accuracy and average error distance, respectively.

In the normal scenario, removing 0% of BSSIDs, RADAR and WinIPS algorithms achieves 100% of floor localization accuracy. RADAR and WinIPS has average error distance for 3D localization 0.43 and 0.54 meters, respectively. However, our *InHit* provides the least error distance. Specifically, its average error distance is 0.12 meters which is less than RADAR and WinIPS.

In the missing-BSSIDs scenarios, the accuracy of RADAR critically decreases. This is because the missing-BSSIDs leads to the sparser vector of incoming query. This affects the *Euclidean distance* value which RADAR utilizes to localize the incoming query. The accuracy of WinIPS slightly decreases for both floor localization and 3D localization because of the uniform scaling process which can relieve the effect of missing-BSSIDs. After integrating with our *MissingHit*, in the case of removing 70% of BSSIDs, the floor localization accuracy in RADAR significantly increases from 66.69% to 99.80% and average error distance significantly decreases from 10.44 to 0.88 meters. Next, the floor localization accuracy in WinIPS increases from 99.89% to 99.92% and average error distance decreases from 1.1 to 0.96 meters. As can be seen in overall, our *MissingHit* can also improve the accuracy of other works for both floor localization and 3D localization.

8) PROCESSING TIME OF THE OVERALL SYSTEM ARCHITECTURE

In this section, we evaluate the processing time of three different indoor localization architectures in order to localize the real incoming queries.

1st Architecture: Only *InHit* is deployed. The indoor localization algorithm compares the incoming query with every signal fingerprint in the database. Thus, the system takes a lot of resources and time to process the incoming query.

TABLE 16. Accuracy of the floor localization of RADAR and WinIPS compared to InHit algorithms in missing-BSSIDs scenarios.

Removed BSSIDs	10%	20%	30%	40%	50%	60%	70%
InHit	100	100	100	100	100	100	100
RADAR	99.95	99.86	99.69	99	94.94	82.67	66.69
RADAR with *MissingHit*	100	100	99.98	99.94	99.97	99.97	99.80
WinIPS	100	100	100	99.95	99.95	99.86	99.90
WinIPS with *MissingHit*	100	100	100	99.98	99.97	99.97	99.92

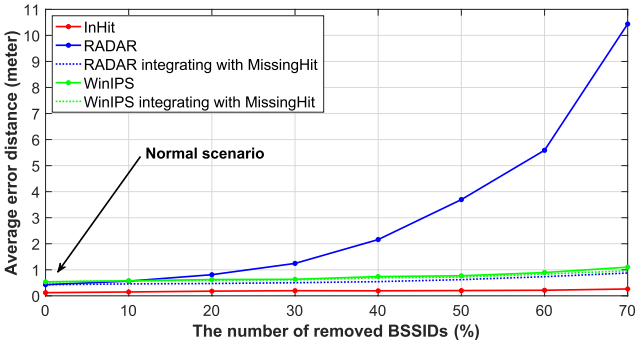


FIGURE 15. Average error distance of the 3D localization of RADAR and WinIPS compared to InHit algorithms in missing-BSSIDs scenarios.

2nd Architecture: **ExtHit** and *InHit* are deployed. Refer to Figure 2 for the 2nd architecture. This is our proposed architecture in which *ExtHit* identifies the incoming query whether it is outdoor or located in a specific building. Then, *InHit* can use this information to reduce searching space. Thus, the system takes less resources and time to process the incoming query compared to the 1st architecture.

3rd Architecture: the system in [26] is deployed with *InHit*. Since the system in [26] contains only an area classification algorithm, we integrated our *InHit*, an indoor localization algorithm, with the system in [26] to complete the indoor localization system. Thus, the system in [26] is used to reduce searching space for *InHit*.

We used the 2nd dataset to create the fingerprint and the 5th dataset which was gathered from the real users' queries to test the overall processing time for each architecture.

The 5th dataset contains 609,277 records of queries from users. 405,969 records are invalid since they contain no BSSIDs. This is because the users did not turn on their Wi-Fi interfaces. In this evaluation, we used the remaining 203,308 records which are valid.

The cumulative processing time spent in each architecture is shown in Figure 16. Table 17 summarizes the cumulative processing time and the number of queries that the system detects as an indoor query. As can be seen, using *ExtHit* to limit searching space for *InHit* can significantly reduce the overall processing time from 4,446 to 227 seconds. However, the result shows that deploying the area classification system in [26] cannot significantly reduce the overall processing time. This is because, in the large-scale area, there are a lot of

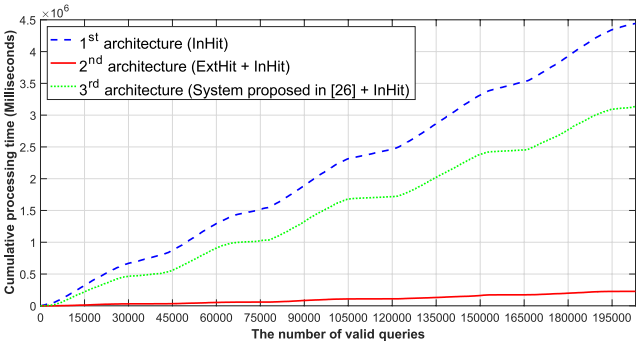


FIGURE 16. Cumulative processing time spent in three different architectures.

TABLE 17. The cumulative processing time and localization result of each architecture.

Architecture	Cumulative processing time (second)	The number of queries that the system detects as an indoor query
1st architecture (*InHit*)	4,446	39,573
2nd architecture (*ExtHit* integrating with *InHit*)	227	27,438
3rd architecture (System in [26] integrating with *InHit*)	3,135	12,634

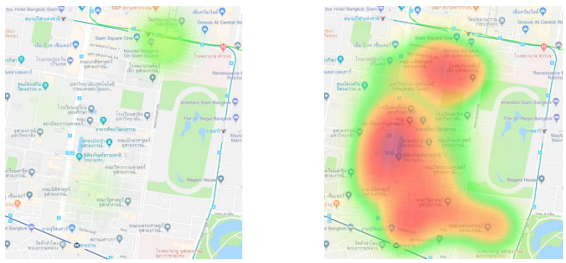


FIGURE 17. Heatmap visualization of users' queries at different time. (a) Heat map of the valid queries during nighttime. (b) Heat map of the valid queries during daytime.

unique access points. This leads to a large dimension of data and long feature extraction processing time.

Moreover, according to Table 17, the number of queries that the 1st architecture detects as indoor was approximately 40,000 queries. Most of these queries were actually sent from outdoor but *InHit* returned as an indoor location. This false detection leads to wasteful resource consumption and processing time. This shows that the deployment of area classification is highly necessary.

In Figure 16, we observed that the slopes of all architectures change periodically. Specifically, the slopes decrease at the 30,000th valid query and increase at the 45,000th valid query. This is because the higher-slope part was the queries obtained during the daytime when the exhibition was opening. The lower-slope part was the queries obtained during the

nighttime when the exhibition was closing. Thus, the number of BSSIDs in the user's queries depended on the period of time. In the daytime, the average number of BSSIDs in the user's queries was 44 BSSIDs while in the nighttime, it was only 7 BSSIDs. Thus, the processing time of the algorithm was significantly different between daytime and nighttime.

To support the above-mentioned summarization, we visualized the GPS information from users' queries by using a heat map. Figure 17(a) and 17(b) show the heat map of the valid queries during nighttime and daytime, respectively.

V. CONCLUSION

In this article, we propose an adaptive indoor localization system for a large-scale area. The system consists of three parts: an area classification algorithm, a fingerprint-based indoor localization algorithm and a missing-BSSID detector algorithm. (1) the area classification algorithm consists of three modules. First, the unknown-BSSID filtering module filters out the unknown BSSIDs in the incoming query. Second, the indoor/outdoor identification module identifies whether the user is indoor or outdoor. It can achieve up to 93.27% accuracy. This module does not require outdoor fingerprints, thus does not need man power to collect fingerprints. Third, the building identification module identifies at which building the user is located. It can achieve up to 100% accuracy. The algorithm is robust to heterogeneous mobile devices and the changing environment. Moreover, the algorithm is compatible with other fingerprint-based indoor localization algorithms. (2) The fingerprint-based indoor localization algorithm achieves up to 100% accuracy for floor localization and has average error distance 0.12 meters for localization. (3) The missing-BSSID detector algorithm detects the missing BSSIDs in the incoming query and updates the sampling database to improve the accuracy of other parts. It can achieve up to 98.03% detection rate. Besides, the results show that using our area classification algorithm to limit searching space for our indoor localization algorithm can significantly reduce the overall processing time compared to the existing work. Moreover, our missing-BSSID detector algorithm can effectively work together with our area classification algorithm and our indoor localization algorithm. It can also work with other existing area classification algorithms and indoor localization algorithms in order to improve accuracy of the overall system in the changing environment.

REFERENCES

[1] M. S. Bargh and R. D. Groote, "Indoor localization based on response rate of Bluetooth inquiries," in *Proc. 1st ACM Int. Workshop Mobile Entity Localization Tracking GPS-Less Environ. (MELT)*, 2008, pp. 49–54.

[2] R. Faragher and R. Harle, "Location fingerprinting with Bluetooth low energy beacons," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 11, pp. 2418–2428, Nov. 2015.

[3] Y. Gu and F. Ren, "Energy-efficient indoor localization of smart hand-held devices using Bluetooth," *IEEE Access*, vol. 3, pp. 1450–1461, 2015.

[4] R. Giuliano, G. C. Cardarilli, C. Cesarini, L. D. Nunzio, F. Fallucchi, R. Fazzolari, F. Mazzenga, M. Re, and A. Vizzarri, "Indoor localization system based on Bluetooth low energy for museum applications," *Electronics*, vol. 9, no. 6, p. 1055, Jun. 2020.

[5] M. Sugano, T. Kawazoe, Y. Ohta, and M. Murata, "Indoor localization system using RSSI measurement of wireless sensor network based on ZigBee standard," *Wireless Opt. Commun.*, vol. 538, pp. 1–6, Jul. 2006.

[6] S. Holm, "Hybrid ultrasound-RFID indoor positioning: Combining the best of both worlds," in *Proc. IEEE Int. Conf. RFID*, Apr. 2009, pp. 155–162.

[7] Y. Brian, S. Wu, H. R. Wu, and K. Zhang, "Overview of RFID-based indoor positioning technology," in *Proc. GSR*, 2012, pp. 1–10.

[8] Y. Ma, B. Wang, S. Pei, Y. Zhang, S. Zhang, and J. Yu, "An indoor localization method based on AOA and PDOA using virtual stations in multipath and NLOS environments for passive UHF RFID," *IEEE Access*, vol. 6, pp. 31772–31782, 2018.

[9] M. El-Absi, A. A. Abbas, A. Abuelhaija, F. Zheng, K. Solbach, and T. Kaiser, "High-accuracy indoor localization based on chipless RFID systems at THz band," *IEEE Access*, vol. 6, pp. 54355–54368, 2018.

[10] E. Hatem, S. Abou-Chakra, E. Colin, J.-M. Laheurte, and B. El-Hassan, "Performance, accuracy and generalization capability of RFID tags' constellation for indoor localization," *Sensors*, vol. 20, no. 15, p. 4100, Jul. 2020.

[11] K. Liu, X. Liu, and X. Li, "Guoguo: Enabling fine-grained indoor localization via smartphone," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2013, pp. 235–248.

[12] A. B. M. Musa and J. Eriksson, "Tracking unmodified smartphones using Wi-Fi monitors," in *Proc. 10th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2012, pp. 281–294.

[13] C. Luo, L. Cheng, M. C. Chan, Y. Gu, J. Li, and Z. Ming, "Pallas: Self-bootstrapping fine-grained passive indoor localization using WiFi monitors," *IEEE Trans. Mobile Comput.*, vol. 16, no. 2, pp. 466–481, Feb. 2017.

[14] W. Liu, H. Chen, Z. Deng, X. Zheng, X. Fu, and Q. Cheng, "LC-DNN: Local connection based deep neural network for indoor localization with CSI," *IEEE Access*, vol. 8, pp. 108720–108730, 2020.

[15] T. Koike-Akino, P. Wang, M. Pajovic, H. Sun, and P. V. Orlik, "Fingerprinting-based indoor localization with commercial mmWave WiFi: A deep learning approach," *IEEE Access*, vol. 8, pp. 84879–84892, 2020.

[16] K. Khaoampai, K. Na Nakorn, and K. Rojviboonchai, "FloorLoc-SL: Floor localization system with fingerprint self-learning mechanism," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 11, Nov. 2015, Art. no. 523403.

[17] H. Ye, T. Gu, X. Zhu, J. Xu, X. Tao, J. Lu, and N. Jin, "FTrack: Infrastructure-free floor localization via mobile phone sensing," in *Proc. Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2012, pp. 2–10.

[18] M. Zhang, L. Pei, and X. Deng, "GraphSLAM-based crowdsourcing framework for indoor Wi-Fi fingerprinting," in *Proc. 4th Int. Conf. Ubiquitous Positioning, Indoor Navigat. Location Based Services (UPINLBS)*, Nov. 2016, pp. 61–67.

[19] M. Zhang, Y. Wen, J. Chen, X. Yang, R. Gao, and H. Zhao, "Pedestrian dead-reckoning indoor localization based on OS-ELM," *IEEE Access*, vol. 6, pp. 6116–6129, 2018.

[20] R. Górak, M. Luckner, M. Okulewicz, J. Porter-Sobieraj, and P. Wawrzyniak, "Indoor localisation based on GSM signals: Multistorey building study," *Mobile Inf. Syst.*, vol. 2016, pp. 1–17, Apr. 2016.

[21] A. Varshavsky, A. LaMarca, J. Hightower, and E. D. Lara, "The SkyLoc floor localization system," in *Proc. 5th Annu. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2007, pp. 125–134.

[22] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proc. 9th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2011, pp. 141–154.

[23] K. P. Subbu, B. Gozick, and R. Dantu, "LocateMe: Magnetic-fields-based indoor localization using smartphones," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, pp. 1–27, Sep. 2013.

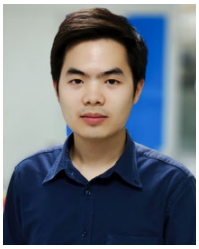
[24] S. Lee, S. Chae, and D. Han, "ILoA: Indoor localization using augmented vector of geomagnetic field," *IEEE Access*, vol. 8, pp. 184242–184255, 2020.

[25] Q. Niu, T. He, N. Liu, S. He, X. Luo, and F. Zhou, "Mail: Multi-scale attention-guided indoor localization using geomagnetic sequences," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 4, no. 2, pp. 1–23, 2020.

[26] S. He, J. Tan, and S.-H.-G. Chan, "Towards area classification for large-scale fingerprint-based system," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Sep. 2016, pp. 232–243.

[27] T. Vongsuteera, K. N. Nakorn, and K. Rojviboonchai, "Floor localization algorithm utilizing different order of access point from Wi-Fi signal fingerprint," in *Proc. 13th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, Jun. 2016, pp. 1–6.

[28] K. Khaoampai, K. N. Nakorn, and K. Rojviboonchai, "Low complexity floor localization algorithm for mobile phone," in *Proc. 11th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, May 2014, pp. 1–6.
[29] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proc. 3rd Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2005, pp. 205–218.
[30] 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.
[31] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. IEEE INFOCOM Conf. Comput. Commun. 19th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 2. Piscataway, NJ, USA: IEEE Press, 2000, pp. 775–784.
[32] H. Zou, M. Jin, H. Jiang, L. Xie, and C. J. Spanos, "WinIPS: WiFi-based non-intrusive indoor positioning system with online radio map construction and adaptation," *IEEE Trans. Wireless Commun.*, vol. 16, no. 12, pp. 8118–8130, Dec. 2017.
[33] R. Montoliu, E. Sansano-Sansano, A. Gascó, O. Belmonte, and A. Caballer, "Senior monitoring: A real case of applying a WiFi fingerprinting-based indoor positioning method for people monitoring," in *Proc. IPIN (Short Papers/Work-Prog. Papers)*, 2019, pp. 95–102.
[34] J. Yoo and S. Park, "Fingerprint variation detection by unlabeled data for indoor localization," *Pervasive Mobile Comput.*, vol. 67, Sep. 2020, Art. no. 101219.
[35] O. Hashem, M. Youssef, and K. A. Harras, "WiNar: RTT-based sub-meter indoor localization using commercial devices," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2020, pp. 1–10.
[36] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen, "IODetector: A generic service for indoor outdoor detection," in *Proc. 10th ACM Conf. Embedded Netw. Sensor Syst.*, 2012, pp. 113–126.
[37] V. Radu, P. Katsikouli, R. Sarkar, and M. K. Marina, "A semi-supervised learning approach for robust indoor-outdoor detection with smartphones," in *Proc. 12th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2014, pp. 280–294.
[38] M. Azizyan, I. Constandache, and R. Roy Choudhury, "SurroundSense: Mobile phone localization via ambience fingerprinting," in *Proc. 15th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2009, pp. 261–272.
[39] F. Zafari, A. Gkelias, and K. Leung, "A survey of indoor localization systems and technologies," 2017, *arXiv:1709.01015*. [Online]. Available: http://arxiv.org/abs/1709.01015
[40] S. Sadowski and P. Spachos, "RSSI-based indoor localization with the Internet of Things," *IEEE Access*, vol. 6, pp. 30149–30161, 2018.
[41] F. Gu, X. Hu, M. Ramezani, D. Acharya, K. Khoshelham, S. Valaee, and J. Shang, "Indoor localization improved by spatial context—A survey," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–35, Jul. 2019.
[42] X. Zhu, W. Qu, T. Qiu, L. Zhao, M. Atiquzzaman, and D. O. Wu, "Indoor intelligent fingerprint-based localization: Principles, approaches and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2634–2657, 4th Quart., 2020.



TEERAPAT VONGSUTEERA received the B.Eng. degree in computer engineering from Chulalongkorn University, Thailand, in 2015, where he is currently pursuing the Ph.D. degree in computer engineering. His current research interests are indoor localization systems and smart city applications.

KULTIDA ROJVIBOONCHAI (Member, IEEE) received the B.Eng. degree in electrical engineering from Chulalongkorn University, Thailand, in 2000, and the M.Sc. and Ph.D. degrees in frontier sciences from The University of Tokyo, in 2003 and 2006, respectively. During her study in Japan, she received the Monbukagakusho Scholarship from 2000 to 2006 and the Microsoft Research Asia Fellowship in 2004. She was a Researcher with Hitachi, Ltd., Japan, from 2006 to 2008. She is currently a Professor in computer engineering with Chulalongkorn University. Her research interests include vehicular networks, software-defined systems, indoor localization systems, the IoT platforms, ITS services, and smart city platforms and applications.