Smartphone-assisted Automatic Indoor Localization of BLE-enabled Appliances Using BLE and GNSS Signals

Masayuki Fujiwara
Daikin Industries, Ltd.

Tomoya Nakatani
Graduate School of Information Science and Technology, Osaka University

Yiming Tian
Graduate School of Information Science and Technology, Osaka University

Joseph Korpela
Graduate School of Information Science and Technology, Osaka University

Takuya Maekawa
Graduate School of Information Science and Technology, Osaka University

Takahiro Hara
Graduate School of Information Science and Technology, Osaka University

ABSTRACT

Information about indoor locations of intelligent appliances is necessary to provide smart services by collaboration across the appliances. This study proposes a method to automatically estimate the indoor coordinates of Bluetooth Low Energy (BLE)-enabled appliances with the help of *unlabeled* BLE and global navigation satellite system (GNSS) data obtained from a smartphone in the environment (e.g., a smartphone carried by a person working or living in the environment). We analyze GNSS data to detect when the smartphone is located near a window, and then estimate the orientation of the outer wall into which that window is installed. By combining the rough indoor positions of the smartphone with the distances to each appliance estimated by BLE signals, we estimate the absolute coordinates of the appliances. We have evaluated the proposed method in real-world environments and achieved an average error distance of about 3 meters.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous computing**; **Mobile computing**.

KEYWORDS

Context recognition, indoor positioning, BLE signal, GNSS

ACM Reference Format:
Masayuki Fujiwara, Tomoya Nakatani, Yiming Tian, Joseph Korpela, Takuya Maekawa, and Takahiro Hara. 2020. Smartphone-assisted Automatic Indoor Localization of BLE-enabled Appliances Using BLE and GNSS Signals. In *The 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '20), November 18–20, 2020, Virtual Event, Japan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3408308.3427981

1 INTRODUCTION

Many appliances, such as air conditioners, lights, displays, and refrigerators, can now be equipped with wireless modules, such as Bluetooth Low Energy (BLE), enabling smarter services by allowing collaboration across appliances. By setting the indoor coordinates of each appliance in a floor map, we can provide smarter services via the appliances [2, 6, 11, 15, 16, 24]. However, in locations where many appliances are installed, manual setting of the indoor coordinates for each of the appliances can be burdensome. In our company, for example, it takes a few days for several employees to manually set the indoor coordinates of the smart appliances (associating network addresses of air conditioners with the coordinates) in a medium-sized building, incurring substantial costs related to sending the employees to each building. Because there are numerous buildings with no information about the coordinates of appliances all over the world, manual settings of the information can incur astronomical costs.

Once the positions of air conditioners and sensor nodes (e.g., temperature, humidity, and CO2 sensors) in an office are given, for example, we can achieve sub-room-level air quality maintenance in the office by collaboration across the air conditioners. Even when an air conditioner in the office does not work, the other air conditioners can collaboratively work to maintain the temperature around the broken air conditioner. Information about positions of the static appliances can also be used to automatically construct an indoor positioning infrastructure for a smartphone in the environment based on the trilateration by leveraging the distance between the smartphone and each appliance estimated by BLE.

In this study, we propose a new method for automatically estimating the indoor coordinates of BLE-enabled appliances in the coordinate system of a floor map. Here we leverage unlabeled BLE and global navigation satellite system (GNSS) data that is automatically obtained from a smartphone in that environment (e.g., a smartphone carried by a person living or working in the environment). In brief, we detect when the smartphone is near a window and then roughly estimate the smartphone's position as being along that window's wall. In addition, we estimate appliance-to-appliance distances and appliance-to-smartphone distances using BLE signal strengths. We then estimate the indoor coordinates of the appliances by fusing these information.

In detail, we employ GNSS data to detect when the smartphone is located close to a window. This is done based on our assumption that "signal strengths from GPS satellites are strongest at positions close to a window." We then estimate the orientation of the outer wall into which that window is installed. This is done based on our

BuildSys '20, November 18–20, 2020, Virtual Event, Japan

ACM ISBN 978-1-4503-8061-4/20/11…$15.00
https://doi.org/10.1145/3408308.3427981

assumption that "signal strengths from a GPS satellite at a particular azimuth are strongest near the outer wall whose orientation aligns with that azimuth" as shown in Fig. 1. In both cases, we conduct supervised machine learning using an environment-independent approach in which the labeled training data is collected in environments other than the environment of interest.

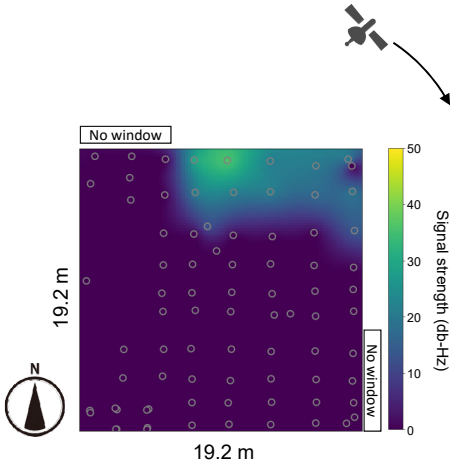


Figure 1: A heat map generated from signals observed from a GPS satellite. Each gray circle indicates an observation point. During data collection, the elevation angle of the satellite changed from 26 to 29 degrees (above the horizon) while the azimuth of the satellite changed from 49 to 59 degrees (relative to north).

Having detected that the smartphone is near a window and determined the orientation of the nearest wall, we then estimate the distances between each appliance and the smartphone's position using BLE signals. Finally, we compute the absolute coordinates of the appliances by feeding the above estimated information into an optimization solver. Specifically, we optimize the positions of the appliances and the smartphone to best fit (i) the estimated appliance-to-appliance distances and (ii) the estimated appliance-to-smartphone distances, with the smartphone's position constrained to any point along the wall in the floor map that matches our detected wall's orientation. Note that, the number of parameters to optimize in this problem is large, which makes it difficult to avoid falling into a local minimum. Therefore, we initialize the optimization parameters based on the estimated relative coordinates of the appliances, which are close enough to the final results (absolute coordinates) to allow conversion to the final results by simply shifting and rotating the relative coordinates.

Contributions: (i) To the best of our knowledge, this is the first method proposed for automatically estimating the indoor coordinates of BLE-enabled appliances using *unlabeled* BLE and GNSS data. Our idea of leveraging information about window-side and window orientation acquired from GNSS signals is new. We design a novel pipeline of processing and fusing multi-modal sensor data as well as of optimization procedures that enables to avoid falling into a local minimum. (ii) To our knowledge, this is the first study for detecting when a smartphone is located close to a window as well as estimating the orientation of the outer wall nearest to the smartphone based on GNSS data.

2 RELATED WORK

2.1 Indoor positioning with RF signals

RF-based fingerprinting approach requires the construction of a database consisting of RSSI vectors matched with indoor coordinates [5, 12, 21], with a significant cost incurred for the site survey needed to construct the database. Meanwhile, the trilateration approach uses the distance between a signal receiver (such as a smartphone) and a reference transmitter (such as a BLE tag) [22, 23, 25], which also incurs a significant cost due to the site survey needed to set the coordinates of reference transmitters. In contrast, this study proposes a method for automatically estimating the indoor positions of BLE-enabled appliances.

2.2 Distance estimation with RF signals

Because BLE (and Wi-Fi) signals are a type of electromagnetic wave, the aforementioned trilateration studies take advantage of the electromagnetic attenuation of the signals in order to estimate the distance between the transmitter and the receiver based on the signal strength. In contrast, Nakatani et al. [17] used machine learning to estimate the distance between two signal receivers (smartphones) based on the signals received from ambient Wi-Fi access points (APs).

2.3 Indoor positioning with GPS

Several studies have investigated indoor positioning using GPS. Kjærgaard et al. [10] investigated the positioning performance of a dedicated GPS receiver and mobile phones in indoor environments. While the positioning error of the dedicated receiver was below 10 meters, the error of GPS receivers embedded in mobile phones was considerable higher. Ochiai et al. [3, 18] achieved indoor positioning using GPS signals based on a fingerprinting approach, which required a site survey to be conducted in the target environment. In contrast, we have developed environment-independent models for window proximity detection and wall orientation classification based on GPS signals.

3 APPLIANCE LOCALIZATION METHOD

3.1 Preliminaries

In this study, we assume each appliance is equipped with a BLE module. We also assume that each appliance emits BLE advertising signals as well as observes BLE signal information from the other BLE appliances. In addition, we assume that a smartphone in the environment of interest automatically collects unlabeled GNSS and BLE signal information with a background process (e.g., a smartphone carried by a person living or working in the environment). The GNSS data consists of the elevation angle, azimuth, and signal strength of each visible satellite observed at about 1 Hz. Furthermore, we assume that rough information about the floor map of the environment of interest (outer shape and orientation) is given. In this study, we define the orientation of a wall to be the angle (relative to north) of the line that runs perpendicular to the surface of that wall. We then use the sensor data (BLE and GNSS data)

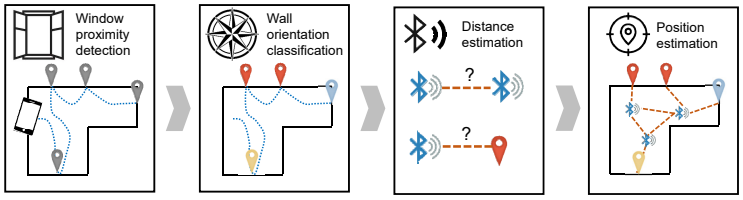


Figure 2: Overview of the proposed method

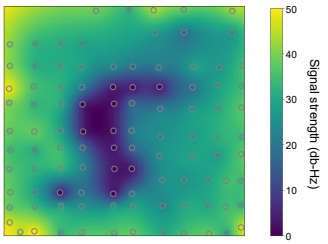


Figure 3: A heat map generated from feature values calculated for each observation point in an environment where the strongest signal strength observed from among all the satellites within a time window is used as the feature. Each gray circle indicates an observation point.

collected by the appliances and smartphone for a given period as inputs to our method.

3.2 Overview

Fig. 2 shows an overview of the proposed method which consists of the following procedures: (i) window proximity detection, (ii) wall orientation classification, (iii) distance estimation, and (iv) position estimation. In the window proximity detection procedure, we detect when the smartphone is located close to a window. In the wall orientation classification procedure, we estimate the orientation of the outer wall into which the window detected in the previous procedure is installed. In the distance estimation procedure, we estimate the appliance-to-appliance distances as well as the appliance-to-smartphone distances. In the position estimation procedure, we aggregate the above distance estimates and locate the appliances by solving an optimization problem.

3.3 Window proximity detection

Although it is difficult to provide reliable positioning for a GPS receiver (smartphone) in an indoor environment, the receiver can still receive signals from a few GPS satellites when it is within close proximity of a window. The proposed method leverages this fact to perform window proximity detection, with *(window proximity)* referring to an area within d_w-meters from a window. Because the signals received from a GPS satellite depend on various factors such as: the satellite's elevation/azimuth, the indoor position of the receiver, and the presence of surrounding buildings; we first collect real-world data in various environments and then employ supervised machine learning to detect when the smartphone is near a window.

3.3.1 Preprocessing. Given a time window of GPS signals, we first detect satellites with low elevation angles (i.e., average angle lower than d_{el} degrees) and remove them from the window, since such satellites are more likely to include outlying signals caused by surrounding buildings. We then smooth the time series of signal strengths within the time window for each satellite by employing a median filter.

3.3.2 Feature extraction. We extract the following features from each preprocessed time window and concatenate the features to construct a feature vector.

- *Maximum signal strength*: The strongest signal strength observed from among the all satellites within the time window. Fig. 3 shows a heat map generated by calculating this feature for each observation point in an example environment, indicating that the values for this feature seem to be strongest at points near outer walls.

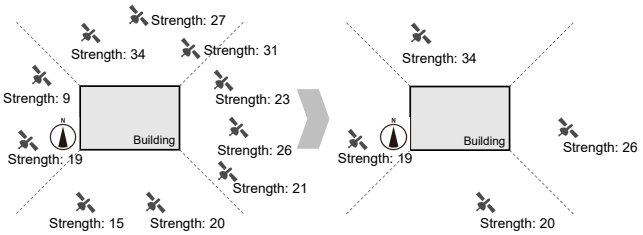


Figure 4: Computing features for window proximity detection: grouping satellites and finding the maximum signal strength.

- *Median signal strength per orientation*: As shown in Fig. 4, we first group satellites within the time window based on their reported azimuths, with satellites assigned to a group if their azimuth is between $o_i - 45$ and $o_i + 45$ degrees, where o_i is the orientation of one of the walls in the environment. We then find the *maximum signal strength* for each group and finally compute the median of those four values and use it as a feature. We compute this feature because in cases where a satellite is at low elevation and aligned well with the nearest wall's orientation, it is possible to receive high signal strengths even when not within d_w-meters from the window. In such cases, it is helpful to check the signal strengths from orientations other than the maximum orientation, since they tend to drop more quickly with increasing d_w.
- *Median signal strength for the maximum orientation*: Just as with *median signal strength per orientation*, we first group the satellites within the time window based on their reported azimuths. However, in this case, we find the group that contains the satellite that corresponds to the time window's *maximum signal strength* and then compute the median signal strength for the satellites within that group. We include this feature since we expect the satellites whose azimuth matches the nearest wall's orientation to have the most reliable signals.

3.3.3 Detection. We create our training dataset by labeling feature vectors that were collected in training environments. We label feature vectors in the training dataset as "window-side" when they were collected in an area within d_w-meters from a window, and as "not" otherwise. ($d_w = 0.5$) We then use these labeled vectors to train a binary classifier using the random forest algorithm [1] in

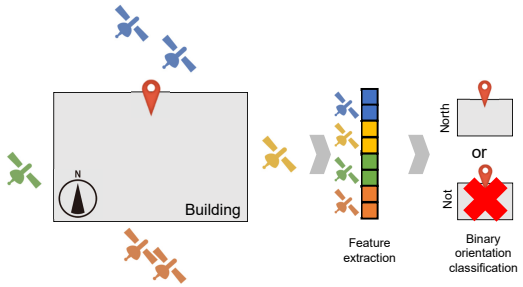


Figure 5: Wall orientation classification. An example of binary classification between the "north" and "not."

scikit-learn v. 0.21 with default settings. The resulting classifier can then be used to classify feature vectors as "window-side" or "not" when used in a testing environment.

3.4 Wall orientation classification

In the previous procedure, we detected when GPS signals were collected within d_w-meters from a window. We next want to estimate the orientation of the outer wall into which that window is installed. This method is designed based on our assumption that when the smartphone is located close to a window, the signals arriving from satellites whose azimuth aligns with the orientation of that window are stronger than those from other satellites. In this method, we prepare a single binary classifier that can determine when the smartphone is located near a wall with some given orientation. Note that, because we prepare only one classifier for orientation classification, we extract a feature vector for each orientation in the environment from each time window. For example, when an environment has walls facing to the north, south, east, and west, we extract four feature vectors from each time window which are fed into the classifier (Figure 5). We then perform one-vs-rest classification for each feature vector and then aggregate the results to determine the final result (i.e., estimated orientation). Using this design allows us to train a single classifier for orientation classification, which can then be used for every orientation in the environment.

3.4.1 Feature extraction. For each time window, we compute m feature vectors corresponding to the m orientations in our environment. The features computed are (i) the maximum signal strength per orientation and (ii) the median signal strength per orientation, with the values for each computed relative to a single orientation of interest o_i. For example, in an environment with four orientations: north, south, east, and west; four feature vectors would be extracted per time window with one of those four composed of features computed relative to north ($o_i = 0$) as follows:
- The maximum/median signal strength among satellites located between $o_i - 90$ and $o_i + 90$ degrees, where o_i corresponds to north, i.e., 0 degrees.
- The maximum/median signal strength among satellites located between $(o_i - 90) - 90$ and $(o_i - 90) + 90$ degrees, where $(o_i - 90)$ is a counterclockwise rotation of 90 degrees from o_i, i.e., 270 degrees.
- The maximum/median signal strength among satellites located between $(o_i + 90) - 90$ and $(o_i + 90) + 90$ degrees, where $(o_i + 90)$ is a clockwise rotation of 90 degrees from o_i, i.e., 90 degrees.
- The maximum/median signal strength among satellites located between $(o_i + 180) - 90$ and $(o_i + 180) + 90$ degrees, where $(o_i + 180)$ is the orientation opposite of o_i, i.e., 180 degrees.

We would then concatenate these eight values into a feature vector that the classifier could use to determine if the nearest wall has a northern orientation. We would also compute feature vectors for east $o_i = 90$, west $o_i = 270$, and south $o_i = 180$ in the same way.

3.4.2 Classification. Using the feature vectors extracted above, we then train a random forest classifier using data collected in training environments. When labeling the training data, the feature vectors extracted for an orientation of interest o_i are labeled as "true" when they were collected within d_w-meters of the wall with orientation o_i, and as "false" otherwise. When estimating the orientation for a time window, we compute the class probability for "true" for each orientation o_i using the corresponding feature vector and then select the orientation with the highest probability, which is the final estimated orientation.

3.4.3 Detecting approximate smartphone positions. Using the procedures outlined above, we are able to identify when a smartphone is close to a window along with the orientation of the wall into which that window is installed. Using this information, we can now detect when a smartphone's position is located next to a wall of a given orientation and can then consider its approximate position at that moment to be at any point along the detected wall. After collected several such approximate positions, we then rank the positions based on their class probabilities and select from them the top-k positions, which are used in the following procedures.

3.5 Distance estimation

Both appliance-to-appliance and appliance-to-smartphone distances are estimated based on BLE signals using deep learning. When estimating appliance-to-appliance distances, the neural network is trained using the BLE signals that each appliance receives from the other appliances. When estimating appliance-to-smartphone distances, the neural network is trained using the BLE signals that the smartphone received from appliances.

3.5.1 Appliance-to-appliance distance estimation. Received signal strength greatly depends on environmental factors such as the number of obstacles (e.g., walls), the types of materials in the obstacles, and the ambient temperature. Therefore, in order to achieve environment-independent distance estimation, we train a neural network that is used for distance estimation based on the approach described in Nakatani et al. [17].

When estimating the distance between two appliances (appliance A and appliance B), we start with a time window of BLE signals composed of the signals received by both appliances from all other appliances in the environment. We then create four vectors to represent each time window: the time series of signal strengths received by appliance A from appliance B $s_{A,B}$, the time series of signal strengths received by appliance B from appliance A $s_{B,A}$, a vector composed of the average signal strengths received by appliance A from all other appliances $\boldsymbol{w_A}$, and a vector composed of the average signal strengths received by appliance B from all other appliances $\boldsymbol{w_B}$. In the case of $\boldsymbol{w_A}$ and $\boldsymbol{w_B}$, the vectors are $|\mathcal{A}|$-dimensional vectors, where $\mathcal{A}$ is the set of all appliances in the environment, with

the value corresponding to the current appliance's average signal strength (i.e., appliance A or B) along with any missing values replaced by the constant -100 dBm.

Using the four vectors described above, we then extract several features to use as input for our neural network when estimating the physical distance between appliance A and B. First, we compute the average, maximum, and minimum of $s_{A,B}$ and $s_{B,A}$, which are directly influenced by the physical distance between the two appliances along with any obstacles lying between them. We then we compute the Chebyshev distance between $\boldsymbol{w_A}$ and $\boldsymbol{w_B}$ as follows:

$$\max_{n \in \mathcal{A}} |\boldsymbol{w_A}(b_n) - \boldsymbol{w_B}(b_n)|, \tag{1}$$

where $\mathcal{A}$ is a set of all appliances in the environment and $\boldsymbol{w_A}(b_n)$ is the strength of the signal observed by appliance A from the n-th appliance. Using the Chebyshev distance between these vectors gives us an indirect measure of the distance between the two appliances that is reported to be robust against environmental differences [17], which is based on the idea that the Chebyshev distance between two signal strength vectors is related to the physical distance between the locations where the vectors were observed [17].

The neural network used for appliance-to-appliance distance estimation is composed of four densely connected layers, along with an output layer that outputs estimated distances. The densely connected layers each consist of 32 nodes and use the ReLU activation function. He initialization [7] is used to initialize the network weights, Adam [9] is used for optimization, dropout [19] is used for regularization, and mean squared error is used as the loss function. The network is trained using mini-batch training using a batch size of ten, with batch normalization [8] applied. We used Keras v. 2.3.1 to implement the network.

3.5.2 Appliance-to-smartphone distance estimation. When estimating the distance between an appliance and a smartphone (appliance A and smartphone S), we start with the time series of signal strengths received by the smartphone from that appliance $s_{S,A}$. We then extract two features from $s_{S,A}$: the average and the variance. These features are then used to train a neural network for distance estimation, with all parameters for this network matching those used during appliance-to-appliance distance estimation, except that the dense layers each consist of only 16 nodes. Note that since appliance-to-smartphone distances are only useful when we have an approximate position for the smartphone, these distances are only estimated when the smartphone is located at one of the top-k approximate positions described in Section 3.4.3.

3.6 Position estimation

Having estimated the distances between each of the appliances along with distances between each of the appliances and the smartphone, we can finally estimate the absolute coordinates of the appliances in the environment. As shown in Fig. 6, we first estimate the relative coordinates of the appliances and then use those results to estimate the absolute coordinates of the appliances.

3.6.1 Relative position estimation. Our method for estimating relative positions consists of two phases: (i) initialization and (ii) optimization. In the initialization phase, we use a trilateration-based method to estimate initial values for the relative coordinates of the appliances. In the optimization phase, we then optimize these initial coordinate values based on the estimated appliance-to-appliance distances.

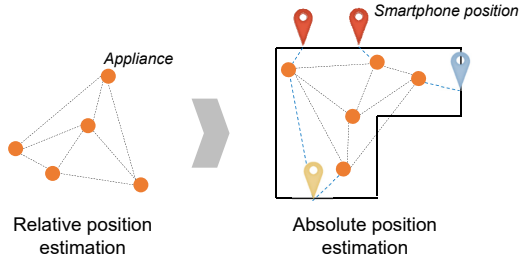


Figure 6: Procedures of position estimation. We estimate the relative coordinates of the appliances and then use those results to estimate the absolute coordinates of the appliances with the help of approximate smartphone positions.

[Initialization]: In this phase, we use the estimated appliance-to-appliance distances to generate initial values for the relative coordinates of the appliances based on trilateration. Fig. 7 shows the procedures used during this phase. In procedure 1, we attempt to find two appliances that are centrally located amongst all the appliances. By doing so, we hope to find two appliances that have good reception of the other appliances' signals. Since appliance-to-appliance distances estimated from the signals greatly affect the accuracy of trilateration, starting with such appliances can improve the final performance of the initialization. We start with the average signal strength vectors computed for each appliance during appliance-to-appliance distance estimation (e.g., $\boldsymbol{w_A}$ for appliance A and $\boldsymbol{w_B}$ for appliance B) and average these vectors to give us a vector representing the average signal strengths received from all appliances $\boldsymbol{w_{avg}}$. We then find the appliance whose signal strength vector is the most similar to $\boldsymbol{w_{avg}}$, which we will refer to as the 1st appliance. We also find the appliance whose signal strength vector is the second most similar to $\boldsymbol{w_{avg}}$, which we will refer to as the 2nd appliance. These two appliances are then used as our two centrally located appliances.

In procedure 2, we set the position of the 1st appliance as the origin of the coordinate system. We then orient the axes so the 2nd appliance is positioned on the positive x-axis, so that the coordinates of the 2nd appliance are $(d(b_1, b_2), 0)$, where b_1 and b_2 indicate the 1st and 2nd appliances, respectively, and $d(b_1, b_2)$ is the estimated distance between the 1st and 2nd appliances based on BLE signals.

In procedure 3, we find the appliance that has the shortest average distance from the 1st and 2nd appliances and select it as the 3rd appliance. We use the shortest average distance here since we assume that the accuracy of distance estimates is inversely proportional to the distance [17]. We then determine the coordinates of the 3rd appliance based on the intersection points of the circles centered on the 1st and 2nd appliances with the radii of $d(b_1, b_3)$ and $d(b_2, b_3)$, respectively.

In procedure 4, we iteratively determine the coordinates of the remaining appliances. We first randomly select an appliance (the n-th appliance) whose coordinates have not yet been determined. We

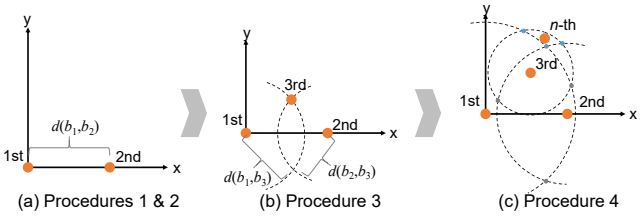


Figure 7: Method for initializing coordinates during relative position estimation. (a) Two centrally located appliances (the 1st and 2nd appliances) are found and the coordinate system is formed. (b) The coordinates of the 3rd appliance are determined based on its distances from the 1st and 2nd appliances. (c) The coordinates of the n-th appliance are determined based on its distances from nearby appliances. In this example, the nearby appliances are the 1st, 2nd, and 3rd appliances. We select three intersection points (small blue dots) that are located close to each other and calculate the centroid of these intersection points as the coordinates for the n-th appliance.

then select three appliances from the appliances whose coordinates have already been determined that have the top-3 shortest distances from that n-th appliance. We finally determine the coordinates of the n-th appliance by using trilateration based on its distances from those three appliances.

[Optimization]: In the second phase, we optimize the coordinates of the appliances generated during the initialization phase to minimize the distance errors between the appliance-to-appliance distances that are based on BLE signals and the appliance-to-appliance distances that are based on the Euclidean distances between the relative coordinates. In addition, when performing optimization, we also take into account the confidence of each BLE-based appliance-to-appliance distance. For example, assume that the BLE-based appliance-to-appliance distance between the n-th and m-th appliances is $d(b_n, b_m)$ and the Euclidean distance between the coordinates of the n-th and m-th appliances is $Eu(p_n, p_m)$, where p_n represents the coordinates of the n-th appliance. We would then optimize p_n and p_m to minimize the error between $d(b_n, b_m)$ and $Eu(p_n, p_m)$ while taking into account the confidence of the BLE-based distance estimate. Specifically, we minimize the following objective function using the L-BFGS-B method [26]:

$$\sum_{n,m\in\mathcal{A}} w_{n,m}\left(d(b_n,b_m) - Eu(p_n,p_m)\right)^2, \tag{2}$$

where $w_{n,m}$ is the confidence of the BLE-based distance between the n-th and m-th appliances, which is inversely proportional to the distance since the errors in distance estimates are expected to increase as the distance increases.

3.6.2 Absolute position estimation. Having calculated the relative coordinates of the appliances, we now combine the relative coordinates with our approximate smartphone positions and appliance-to-smartphone distances in order to transform the relative coordinates into absolute coordinates. We do so by optimizing both the appliance positions and the smartphone positions given several constraints, with the appliance positions initialized by shifting the relative coordinates so that the centroid of the coordinates is in the center of the environment and with the smartphone positions initialized by randomly selecting a position along the wall from within each of the top-k approximate positions selected in Section 3.4.3. The main constraints used during optimization are that the appliance positions are constrained to be within the walls of the environment and that the smartphone positions are constrained to fall on the wall that corresponds to their approximate position. When optimizing the coordinates, we minimize the following objective function:

$$\sum_{n,m\in\mathcal{A}} w_{n,m}\left(d(b_n,b_m) - Eu(p_n,p_m)\right)^2 + \sum_{n\in\mathcal{A},i\in\mathcal{S}} w_{n,i}\left(d(b_n,s_i) - Eu(p_n,p_i)\right)^2, \tag{3}$$

where $\mathcal{S}$ is the set of approximate smartphone positions detected using GNSS signals, s_i represents the i-th smartphone position, and p_i represents the coordinates of the i-th smartphone position.

Optimization is done in two phases: a rough alignment phase and a fine tuning phase. In the rough alignment phase, we minimize the objective function by rotating and shifting the relative coordinates of the appliances as well as by adjusting the smartphone positions. This is done twice, with the relative coordinates of the appliances flipped when initializing the second attempt and with the results with the smallest error selected from the two attempts. In the fine tuning phase, we minimize the objective function by adjusting both the absolute coordinates of the appliances and the coordinates of the smartphone positions using the results from the first phase as our initial values.

Table 1: Experimental environments. Float shows float glass. ALC stands for autoclaved lightweight aerated concrete.

Env.	Size	Floor	# BLE	Type	Wall/Window
A	19.2m x 19.2m	1F	13	Office	ALC/float
B	19.2m x 19.2m	2F	11	Office	ALC/float
C	26.8m x 17.5m	7F	13	Office	ALC/heat insulated
D	37.6m x 31.9m	3F	13	Conference	Concrete/float
E	22.2m x 26.2m	3F	13	Multipurpose	ALC/float

4 EVALUATION

4.1 Data set

Our dataset consists of GNSS and BLE data collected in five different environments (Table 1 and Fig. 8). Environments A and B are two floors located in the same office building. This office building is surrounded on three sides by tall buildings (located to the left, right, and above in the figure). Environment C is also an office floor. The windows in C are heat insulated with a metal film that interferes with GNSS signals. The building is neighbored by a tall building on one side (located to the right in the figure). Environment D is located in a convention center and is much larger than the other environments with taller windows than those found in the other environments. Environment E is a multipurpose room. The outer walls of E have metallic barriers installed, which also interfere with GNSS signals. The building is neighbored closely on two sides with tall buildings (located to the left and right in the figure).

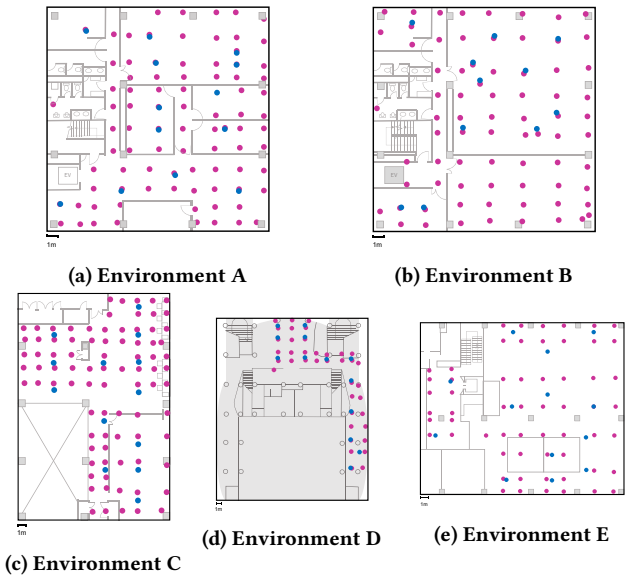


Figure 8: The five environments used during experiments in this study. Blue dots show the locations of BLE-enabled appliances. Purple dots show the locations of observation points.

We installed BLE-enabled appliances (Raspberry Pi 3 with a BLE module) in each environment as shown in Fig. 8. Each appliance emits advertising messages at 10 Hz, and records messages from the other appliances that consist of the BSSID, RSSI, and timestamp. We also collected GNSS and BLE data from multiple smartphone models (Nexus 6P, Nexus 6, Nexus 5X, and Nexus 5) in each environment. GNSS and BLE data were observed at each observation point for 1 minute as shown in Fig. 8. The GNSS data consists of the elevation angle, azimuth, signal strength, and identifier of each visible satellite observed at about 1 Hz. BLE data was collected from all appliances continuously throughout the experiments (about 4 hours).

4.2 Evaluation methodology

The evaluation was conducted using "leave-one-environment-out" cross validation. That is, we iterated through using each environment as the testing environment with the remaining environments used as training environments. Because Environments A and B are located in the same building, when testing on Environment A, we do not use Environment B as a training environment, and vice versa. Each of the models used in our method (distance estimation neural networks and random forests for window proximity detection and window orientation classification) were trained separately on data collected in the training environments from each smartphone model (e.g., Nexus 6P) in order to allow for a comparison of the results based on the model of smartphone.

In order to evaluate the performance of the proposed method, we prepared the following methods:

- Proposed: This is the proposed method.
- W/o initialize: This is based on Proposed, but it does not perform the trilateration-based initialization during relative position estimation.
- W/o weighting: This is based on Proposed, but it does not use the confidence (weight) of distance estimates during the optimization procedures.
- W/o two-phase: This is based on Proposed, but it does not perform the full two-phase optimization process during absolute position estimation. Specifically, this method does not perform the rough alignment phase where the relative coordinates are rotated and shifted. It only performs the second phase of fine tuning.
- W/o three: This is based on Proposed, but combines all three of the above restrictions, i.e., no trilateration-based initialization, no confidence values used during optimization, and only the fine tuning phase used during the initialization process in absolute position estimation.

We evaluated the above methods based on the mean absolute error (MAE) of the position estimates for the appliances.

4.3 Results

4.3.1 Positioning accuracy of the proposed method. Fig. 9 shows the MAEs for each of the five methods for each of the five environments when using a Nexus 6P. In addition, Fig. 10 shows the estimated positions of appliances for the proposed method compared to their ground truth positions. Surprisingly, the proposed method achieved a positioning error of only about 2-3 meters in multiple environments. We believe that an error of about 2-3 meters is close to the lower bound of RSSI-based positioning methods, with this error being similar to that of Wi-Fi-based fingerprinting methods [4, 14], which require a site survey. In contrast, our method does not require any labeled training data to be collected in the target environment. Fig. 10 also illustrates how that the proposed method could accurately estimate positions for each appliance while preserving their relative positioning. However, the positioning errors for some of the appliances, including some that were close to windows, were poor. We believe that these larger errors during positioning are due in part to the lower density of appliances in the areas surrounding those appliances, which increases the distances used when positioning the appliances via BLE signals which in turn increases the errors in the distance estimates (see Section 4.3.9 for more information).

The errors in Environment D are larger than the errors for other environments because Environment D is much larger than the other environments, which results in larger appliance-to-appliance distances being used during positioning. The errors in Environment C ($k = 5$) are larger than in other environments because we were only able to obtain smartphone positions for two opposite orientations, i.e., the right and left sides in Fig. 8 (c), due to the heat insulated windows that interfered with GPS reception. In such cases, we can only align the relative coordinates based on one axis of orientation (e.g., the east-west axis of orientation), making it impossible to disambiguate between two versions of the coordinates that have been flipped perpendicularly to that axis (e.g., with the north and south halves flipped about the east-west axis). A possible solution for this problem is to increase k until we are able to find smartphone positions from perpendicular axes of orientation (e.g., east and north orientations), however increasing k has the drawback of possibly including positions with lower confidence which could

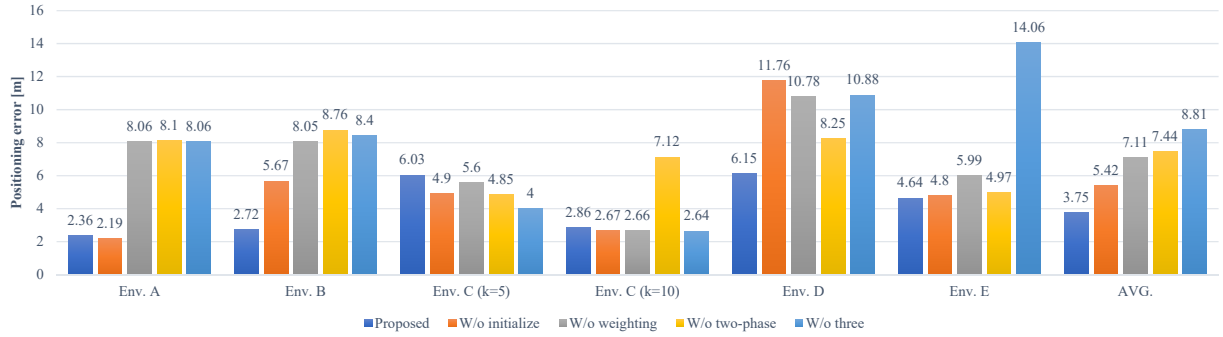
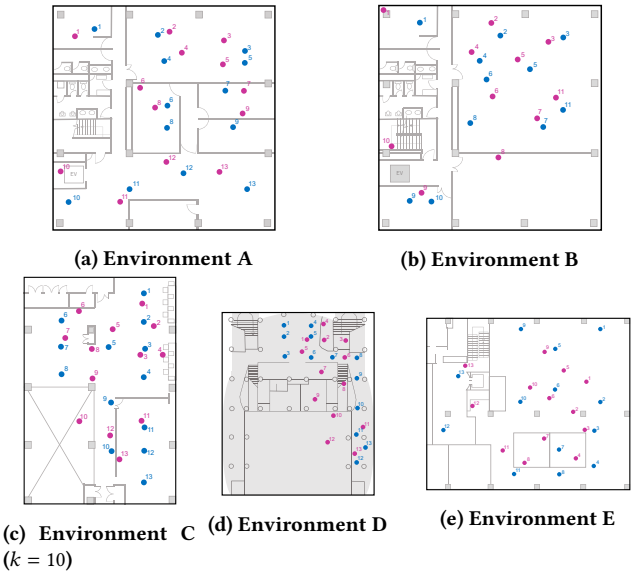


Figure 9: The MAEs for appliance positioning for each of the five methods when using a Nexus 6P. Averages based on these values are calculated using $k = 5$ for Environments A, B, D, and E; and using $k = 10$ for Environment C.



(a) Environment A **(b) Environment B**

(c) Environment C ($k = 10$) **(d) Environment D** **(e) Environment E**

Figure 10: The positioning results for the proposed method when using a Nexus 6P. The blue dots show ground truth positions while the purple dots show estimated positions. Estimates can be matched to their ground truth positions using the included identifiers (e.g, 11).

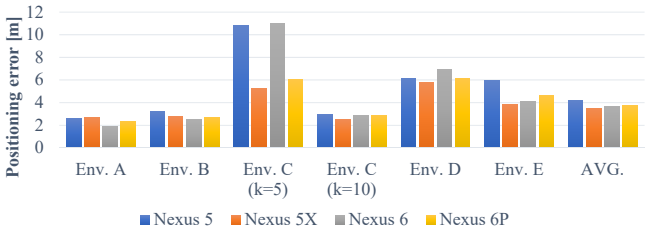


Figure 11: The MAEs for appliance positioning by Proposed for each model of smartphone. Averages based on these values are calculated using $k = 5$ for Environments A, B, D, and E; and using $k = 10$ for Environment C.

degrade the overall quality of the positioning results. Fig. 9 also provides results where k has been increased to remove the ambiguity in Environment C ($k = 10$), where we can see a significant improvement.

Fig. 9 also shows the MAEs for W/o three. As shown in these results, the MAE for W/o three is larger than that of Proposed by about 5.1 meters on average. However, the error for W/o three in Environment C is smaller than that for Proposed. This is because the initial values used during the optimization process for absolute positioning were already similar to the actual (relative) appliance positions. However, the performance of W/o three is not stable.

4.3.2 Contribution of initialization during relative position estimation. As shown in Fig. 9, the MAE of W/o initialize is larger than that of Proposed by about 1.7 meters on average, which may indicate that the additional steps taken to initialize the positions to avoid local minima during relative position optimization are needed for larger environments.

4.3.3 Contribution of weighting during optimization. As shown in Fig. 9, the MAE of W/o weighting is larger than that of Proposed by about 3.4 meters on average. These results illustrate the importance of considering the confidence of the distance estimates, since the errors in the estimates increase with the actual distances (see Section 4.3.9 for details) and can greatly affect the positioning results.

4.3.4 Contribution of two-phase optimization during absolute position estimation. Fig. 9 includes the MAEs for W/o two-phase. The MAE for W/o two-phase is larger than that of Proposed by about 3.7 meters on average, indicating the importance of including the rough alignment phase during initialization.

4.3.5 Results from different models of smartphones. Fig. 11 shows the positioning results for Proposed for the four smartphones. As shown in these results, Nexus 5, which is the oldest product, had the largest average positioning error. This is due to the Nexus 5's poor reception of GPS signals, which is discussed in detail below. However, overall the smartphones all performed similarly.

4.3.6 Accuracy of window proximity detection. Fig. 12 shows the classification precision for the "window-side" class during window proximity detection for different values of k, with the precision for

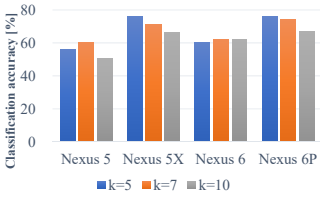


Figure 12: The average classification precision across all five environments for window proximity detection for the "window-side" class for each model of smartphone for different values of k.

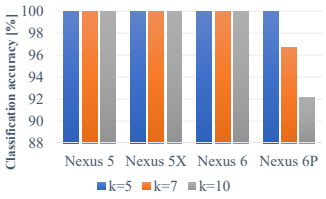


Figure 13: The average classification accuracy across all five environments for wall orientation classification for each model of smartphone for different values of k. These results were calculated using observation points that had been correctly classified as "wall-side."

Table 2: The distance errors (MAE) for appliance-to-appliance distance estimation in each environment.

	Env. A	B	C	D	E
Distance error [m]	2.00	2.63	1.84	2.02	3.64

"window-side" shown here since the positions classified as "window-side" are the ones used during the position estimation phase. As shown in these results, the results for the Nexus 5 are poorer than the results for the other models, which contributed to the Nexus 5's poorer performance during indoor positioning. Furthermore, the results for the other smartphones were also less than perfect. When analyzing the results, we found that many of the erroneous detections came from observation points about one meter away from walls (compared to the $d_w = 0.5$ used to define the "window-side" class). However, the impact of these false positives on the overall positioning performance is limited.

4.3.7 Accuracy of wall orientation classification. Fig. 13 shows the classification accuracy for wall orientation classification for different values of k. As shown in these results, the classification performance was almost perfect. When k increases, positions of Nexus 6P located at corners in the environments were sometimes classified incorrectly.

4.3.8 Accuracy of appliance-to-appliance distance estimation. Table 2 shows the MAE for appliance-to-appliance distance estimation in each environment. Overall, we could achieve an average error of about 2.4 meters across all environments. In the cases of Environments D and E, the higher estimation errors are due in part to the larger actual distances between appliances in those environments.

4.3.9 Accuracy of appliance-to-smartphone distance estimation. Fig. 14 shows the MAEs for appliance-to-smartphone distance estimation for the Nexus 5, 5X, 6, and 6P in each environment. Overall, we can see that we could achieve an average error of about 2-3 meters in these environments. Also note that the errors in Environments D and E are larger than those in the other environments, which is likely due to the increased errors in distance estimates due to the larger distances found in these environments.

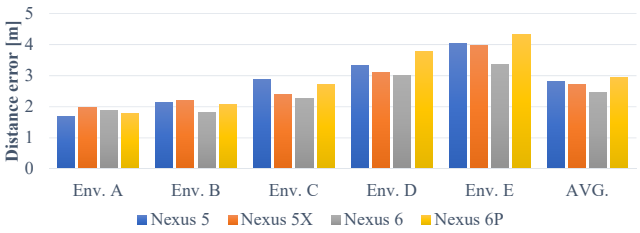


Figure 14: The MAEs for appliance-to-smartphone distance estimation for the four smartphones in each environment.

4.4 Discussion

4.4.1 Device heterogeneity. Here we tested the use of product-independent models, where we trained models for appliance-to-smartphone distance estimation, window proximity detection, and wall orientation classification using data from a Nexus 6P and tested the models using data from the other smartphones. In order to cope with differences in the BLE sensitivity of the devices, we also prepared a linear regression model to convert the BLE signal strengths collected by the testing devices (Nexus 5, 5X, or 6) into estimates of the signal strengths that would have been collected by the training device (Nexus 6P), based on the method described in [13]. Fig. 15 shows the appliance positioning errors for the testing devices when we directly use their BLE data without adjusting their BLE signal strengths using linear regression. Comparing these results to Fig. 11, we can see that in many cases the positioning errors for the product-independent models are not very different from those of the product-dependent models. Based on these results, it appears that because the relative coordinates of the appliances are precisely estimated in the relative position estimation phase, the small errors introduced into the appliance-to-smartphone distance estimation from differing BLE sensitivities did not substantially affect the final appliance positioning results. Fig. 16 shows the appliance positioning errors for the testing devices when we incorporate the linear regression model for adjusting BLE signal strengths, which improved the appliance positioning errors of the product-independent models by 0.2 meters on average.

4.4.2 Number of BLE-enabled appliances. In principle, when the number of appliances in an environment decreases, the positioning error increases because the amount of information used in the optimization process decrease. However, when we reduced the number of appliances from 11 to 5 in Environment B, the positioning error did not change (2.72 vs. 2.73 m). When we drastically reduced the number of appliances from 11 to 3, the the positioning error increased from 2.72 to 5.30 m.

4.4.3 Materials of walls and windows. Because our method relies on GNSS data, the indoor positioning performance degrades in a building with heat insulated windows since the number of detected window-side smartphone positions decreases. In addition, we assume that we cannot receive GNSS signals at places far from windows in a concrete building. However, GNSS signals can be

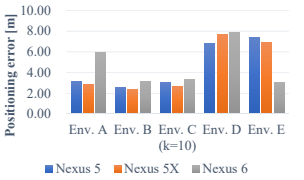


Figure 15: The appliance positioning errors when we use appliance-to-smartphone distance estimation models trained on Nexus 6P data to estimate appliance positions for data collected by Nexus 5, 5X, and 6 smartphones without using linear regression to adjust BLE signal strengths. The average positioning error is 4.57 meters.

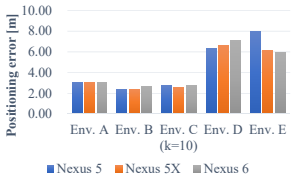


Figure 16: The appliance positioning errors when we use appliance-to-smartphone distance estimation models trained on Nexus 6P data to estimate appliance positions for data collected by Nexus 5, 5X, and 6 smartphones when using linear regression to adjust BLE signal strengths. The average positioning error is 4.30 meters.

observed at such places in a wood building, making it impossible to locate BLE-enabled appliances.

4.4.4 Limitations. A limitation of our experiment is that it relied on unlabeled sensor data collected while the smartphone was stationary at each observation point. In addition, all the experimental environments in this study are concrete buildings, which interfere with GNSS signals.

4.4.5 Future work. As a part of our future work, we plan to include sensor data that also contains non-stationary activities (e.g., walking) as noise. In this case, we can easily eliminate the data collected during non-stationary activities by examining the smartphone's acceleration data, as was proposed in a prior study [20]. In addition, we plan to locate BLE-enabled appliances with limited appliance-to-smartphone distances because these information is obtained only when the appliances emit BLE signals.

5 CONCLUSION

This study proposed a new method for automatically estimating the indoor coordinates of BLE-enabled appliances. The proposed method employs unlabeled data collected by a smartphone to estimate appliance positions without the need for a site survey. We investigated the performance of the proposed method in five real-world environments.

REFERENCES

[1] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[2] Jonghwa Choi, Dongkyoo Shin, and Dongil Shin. 2005. Research and implementation of the context-aware middleware for controlling home appliances. *IEEE Transactions on Consumer Electronics* 51, 1 (2005), 301–306.

[3] Masahiro Fujii and Yoshiaki Mori. 2016. A study on indoor positioning systems based on SkyPlot mask. In *2016 IEEE 5th Global Conference on Consumer Electronics*. IEEE, 1–2.

[4] Yanying Gu, Anthony Lo, and Ignas Niemegeers. 2009. A survey of indoor positioning systems for wireless personal networks. *IEEE Communications Surveys & Tutorials* 11, 1 (2009), 13–32.

[5] Taisei Hayashi, Daisuke Taniuchi, Joseph Korpela, and Takuya Maekawa. 2017. Spatio-temporal adaptive indoor positioning using an ensemble approach. *Pervasive and Mobile Computing* 41 (2017), 319–332.

[6] Mike Hazas, James Scott, and John Krumm. 2004. Location-aware computing comes of age. *Computer* 37, 2 (2004), 95–97.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR* abs/1502.01852 (2015). arXiv:1502.01852 http://arxiv.org/abs/1502.01852

[8] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* abs/1502.03167 (2015). arXiv:1502.03167 http://arxiv.org/abs/1502.03167

[9] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Mikkel Baun Kjærgaard, Henrik Blunck, Torben Godsk, Thomas Toftkjær, Dan Lund Christensen, and Kaj Grønbæk. 2010. Indoor positioning using GPS revisited. In *International conference on pervasive computing*. Springer, 38–56.

[11] Quan Kong, Takuya Maekawa, Taiki Miyanishi, and Takayuki Suyama. 2016. Selecting Home Appliances with Smart Glass Based on Contextual Information. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg, Germany) *(UbiComp '16)*. ACM, New York, NY, USA, 97–108. https://doi.org/10.1145/2971648.2971651

[12] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, et al. 2005. Place lab: Device positioning using radio beacons in the wild. In *International Conference on Pervasive Computing (Pervasive 2005)*. 116–133.

[13] Christos Laoudias, Demetrios Zeinalipour-Yazti, and Christos G Panayiotou. 2013. Crowdsourced indoor localization for diverse devices through radiomap fusion. In *2013 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 1–7.

[14] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. 2007. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 37, 6 (2007), 1067–1080.

[15] Takuya Maekawa, Yutaka Yanagisawa, Yasushi Sakurai, Yasue Kishino, Koji Kamei, and Takeshi Okadome. 2009. Web Searching for Daily Living. In *SIGIR 2009*. 27–34.

[16] Takuya Maekawa, Yutaka Yanagisawa, Yasushi Sakurai, Yasue Kishino, Koji Kamei, and Takeshi Okadome. 2012. Context-aware Web search in ubiquitous sensor environment. *ACM Transactions on Internet Technology (ACM TOIT)* 11, 3 (2012), 12:1–12:23.

[17] Tomoya Nakatani, Takuya Maekawa, Masumi Shirakawa, and Takahiro Hara. 2018. Estimating the Physical Distance Between Two Locations with Wi-Fi Received Signal Strength Information Using Obstacle-aware Approach. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3, Article 130 (Sept. 2018), 26 pages. https://doi.org/10.1145/3264940

[18] Masayuki Ochiai, Masahiro Fujii, Atsushi Ito, Yu Watanabe, and Hiroyuki Hatano. 2014. A study on indoor position estimation based on fingerprinting using GPS signals. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 727–728.

[19] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[20] Masaya Tachikawa, Takuya Maekawa, and Yasuyuki Matsushita. 2016. Predicting location semantics combining active and passive sensing with environment-independent classifier. In *UbiComp 2016*. 220–231.

[21] Daisuke Taniuchi and Takuya Maekawa. 2014. Robust Wi-Fi based indoor positioning with ensemble learning. In *IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2014)*. 592–597.

[22] Andreas Teuber, Bernd Eissfeller, and Thomas Pany. 2006. A two-stage fuzzy logic approach for wireless LAN indoor positioning. In *IEEE/ION Position Location Navigation Symposium*, Vol. 4. 730–738.

[23] Yapeng Wang, Xu Yang, Yutian Zhao, Yue Liu, and Laurie Cuthbert. 2013. Bluetooth positioning using RSSI and triangulation methods. In *IEEE Consumer Communications and Networking Conference (CCNC 2013)*. 837–842.

[24] Tatsuya Yamazaki. 2005. Ubiquitous home: real-life testbed for home context-aware service. In *First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*. IEEE, 54–59.

[25] Junyang Zhou, KM-K Chu, and JK-Y Ng. 2005. Providing location services within a radio cellular network using ellipse propagation model. In *19th International Conference on Advanced Information Networking and Applications (AINA 2005)*, Vol. 1. 559–564.

[26] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)* 23, 4 (1997), 550–560.