applied sciences

MDPI

Article

Enhanced Clustering and Indoor Movement Path Generation from Wi-Fi Fingerprint Data Using Bounding Boxes and Grid Cells

Hong-Gi Shin [1,2], Daesung Lee [3], Chi-Gon Hwang [4,*] and Chang-Pyo Yoon [5,*]

1 School of Robotics, Kwangwoon University, 20, Kwangwoon-ro, Nowon-gu, Seoul 01897, Republic of Korea; ghdrl95@neowiz.com
2 NEOWIZ Corp. 14, Daewangpangyo-ro 645beon-gil, Bundang-gu, Gyeonggi-do, Seongnam-si 13487, Republic of Korea
3 Department of Computer Engineering, Catholic University of Pusan, Busan 46252, Republic of Korea; dslee@cup.ac.kr
4 Department of Computer Engineering, Institute of Information Technology, Kwangwoon University, Seoul 01897, Republic of Korea
5 Department of Computer & Mobile Convergence, Gyeonggi University of Science and Technology, 269, Gyeonggigwagidae-ro, Gyeonggi-do, Siheung-si 15073, Republic of Korea
* Correspondence: duck1052@kw.ac.kr (C.-G.H.); cpyoon@gtec.ac.kr (C.-P.Y.)

Abstract: Recently, various application fields utilizing Wi-Fi fingerprint data have been under research. However, fingerprint data collected from a specific location does not include relevant information, such as continuity. Therefore, most indoor positioning technologies predict the user's location based on location signals collected at specific points within the indoor space, without taking into account the user's movements. Hence, there is a need for technology that improves the accuracy of indoor positioning while moving. This paper proposes a technique to generate movement path data by applying the concepts of "BB" and "Grid Cell" from computer vision to Wi-Fi fingerprint data. This approach represents data points as bounding boxes (BBs), then establishes grid cells and clusters of these BBs to generate adjacency information. Subsequently, movement path data are created based on this information. We utilized the movement path information generated from the dataset as training data for machine learning and introduced an enhanced indoor positioning technology. First, the experiments in this paper assessed the performance of the proposed technology based on the number of paths in the LSTM model. Second, the performance of clustering methods was compared through experiments. Finally, we evaluated the performance of various machine learning models. The experimental results confirmed a maximum accuracy of 94.48% when determining the location based on route information. Clustering performance improved accuracy by up to 3%. In comparative experiments with machine learning models, accuracy improved by up to 2.8%.

Keywords: Wi-Fi fingerprint; calculate movement path; IoU (Intersection over Union); LSTM (Long Short-Term Memory); bounding box (BBs); grid cell

check for updates

Citation: Shin, H.-G.; Lee, D.; Hwang, C.-G.; Yoon, C.-P. Enhanced Clustering and Indoor Movement Path Generation from Wi-Fi Fingerprint Data Using Bounding Boxes and Grid Cells. *Appl. Sci.* **2023**, *13*, 10647. https://doi.org/10.3390/app131910647

Academic Editors: Yun Seop Yu, Kwang-Baek Kim, Dongsik Jo, Hee-Cheol Kim and Jongtae Lee

Received: 27 August 2023
Revised: 20 September 2023
Accepted: 20 September 2023
Published: 25 September 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Recently, Wi-Fi fingerprinting has been used to build indoor positioning systems [1–11]. In this approach, a dataset from a Wi-Fi fingerprinting system uses received signal strength indication (RSSI) information from access points (APs) collected at a specific location point as data to determine location. We calculate the location by comparing the surrounding RSSI values of the location determination point with the previously collected fingerprinting data as input data. However, the RSSI value of a Wi-Fi AP contains irregularities in the noise caused by obstacles, which can lead to erroneous results. Therefore, many studies are conducted to improve the location accuracy [12–14].

Location determination methods are studied for Wi-Fi signals collected by the built-in sensors of smart devices [15]. But the approaches in many studies suffer from the disadvantages of requiring assistive devices and complex configuration [16]. Therefore, another approach to improve location accuracy is to use RSSI along with the mobile terminal's movement path [17–19]. For trajectory estimation along the path of movement, approaches have been proposed that either use dead reckoning trajectories generated from a mounted inertial measurement unit (IMU) or fuse RSSI from Wi-Fi signals and position estimation from a global positioning system (GPS) [18,20]. There is also a study that presents a Wi-Fi RSSI dataset that includes sequentially collected trajectories [19]. Although datasets with trajectories can enhance location accuracy, their practical application in real world environments is challenging due to the substantial additional costs associated with dataset preparation and processing. Additionally, these datasets are limited by the fact that they solely represent the paths travelled during the collection process. In datasets where trajectories are not available (e.g., collected Wi-Fi fingerprints), it is difficult to estimate the mobile device's path of movement. It is possible to create a movement path by using a Recurrent Neural Network (RNN) model that can learn time-series information from a machine learning algorithm [12,13]. RNNs extract features from high-dimensional time-series input data and can be applied to classification and regression problems. RNN models are supervised learning algorithms that can take into account the continuity of the data [21–23]. However, RNNs do not have long-term memory and suffer from the vanishing gradient problem. To solve this problem, variants models such as LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) are used [24–27]. These networks can take into account the continuous movement of the terminal in the indoor localization to more accurately determine its current position or predict its next movement. The use of LSTMs in indoor positioning systems requires sequential RSSI inputs along the terminal's path of movement. However, the datasets provided for most indoor positioning studies do not contain sequential path information.

In this paper, we propose an enhanced Wi-Fi fingerprinting-based method for generating movement path data, building upon previous research. Visualize a Wi-Fi fingerprint dataset and use BB and a grid cell to create clusters that separate indoor location areas. A grid cell is a cluster formed from a dataset. This adjacency was determined by calculating the state transition probability for the BBs within the grid cell, which was then used to generate path data. In the process, we improved upon the problems with parameterization for thresholding in our previous paper. Instead of solely relying on Wi-Fi fingerprinting information, the proposed machine learning model utilizes the movement path data generated through the proposed method. This allows the RSSI of the previous location to influence the positioning of the current location. The performance of the technique was verified through experiments, enhancing the algorithm and refining the movement path generation structure from the previous paper.

1.1. Relatied Works

We describe the techniques required for the indoor localization technique proposed in this thesis. In our previous paper, there is a possibility of errors in cluster formation during the process of clustering adjacent data points, and there is a problem in calculating the distance between clusters that include data points. This can be caused by measuring the distance between clusters and using an arbitrary threshold to connect clusters that are within a certain distance to establish a path. This problem can lead to the establishment of adjacencies between data points that are not actually adjacent to each other, forming incorrect clusters and generating movement paths that have structurally untraversable configurations. To solve the problem, this paper improves the adjacency of BB by representing data points as BBs and completely removing the threshold for adjacency judgment through IoU operations. We also applied grid cells to form clusters of BBs. Below we briefly describe the BB, IoU, and grid cell used in our proposed technique.

1.1.1. IoU (Intersection over Union)

Computational vision is being advanced by CNNs, and various studies are being conducted to detect objects in images. The most basic element used to detect objects in an image is the BB. Object detection uses the BB for the entity and the predicted BB to predict the object in the image and detects the entity by finding and correcting for losses. IoU is the most popular way to compensate for this loss [28]. Extensions to this technique include Generalized IoU (GIoU), Distance IoU (DIoU), and Complete IoU (CIoU), which are basically based on IoU [28,29].

Figure 1 compares the difference between IoU and GIoU presented in the GIoU paper in a situation where the values of l_2-Norm and l_1-Norm are the same. l_2-Norm is the distance between the upper right point (x_1, y_1) of the green box corresponding to BB^{GT} in (a) and the upper right point (x_2, y_2) in (a), and l_1-Norm is the distance between the two boxes in (b), with (x_c, y_c) as the center of the two boxes and w, h as the width and height of the boxes; (a) shows a case where two BBs have the same l_2-Norm, but different IoU and GIoU values, and (b) shows that the same l_1-Norm can result in different IoU and GIoU values.

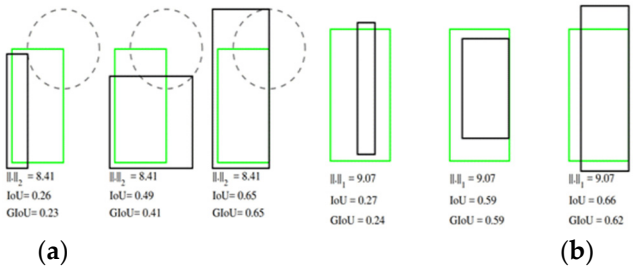


Figure 1. Comparison of IoU and GIoU [28] (**a**) is the case where the 2-Norms of the two BBs are the same, and (**b**) is the case where the 1-Norms are the same.

To express this in a formula, we call the green region A, the black region B, and the region C that minimally combines the two regions. *IoU* and *GIoU* are as follows in Equation (1).

$$IoU = \frac{|A \cap B|}{|A \cup B|}, \; GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|} \tag{1}$$

In this paper, to create a path using Wi-Fi signals, all data points, which are Wi-Fi signals, are treated as images. Each point is represented as a BB, and IoU is used as a criterion for merging and extending neighboring BBs.

1.1.2. YOLO (You Only Look Once)

YOLO [30] is one of the deep learning algorithms for detecting objects in images. YOLO divides the image into grids, and for each grid cell, it predicts the class and BB information of the object it contains. YOLO uses relative position information within grid cells to estimate BB. This means that the coordinates of the BB are expressed as a value relative to the upper left corner of the grid cell. Because of this, YOLO predicts an offset, which is a value representing relative position information. For example, if BB is predicted to be (0.2, 0.3, 0.4, 0.5) within one grid cell, this is the location information relative to the upper left corner of that grid cell. Therefore, in order to calculate the coordinates of the bounding box within the actual image, we need to calculate the absolute coordinate values using the location information of the corresponding grid cells and the predicted offset values. By using this relative position information, YOLO is a technique for predicting the position of objects in an image. In YOLO, the Anchor Box is a box that predefines the shape and size of the BB predicted in each grid cell. Anchor boxes are used to detect objects with different sizes and proportions. The size and shape of the anchor boxes are predefined in YOLO, and these values are set differently depending on the dataset. Each

grid cell can have multiple anchor boxes, which allows to detect objects of different sizes and shapes. By calculating the difference between the predicted BB and the anchor box, YOLO accurately predicts the position and size of the object. Direct location prediction in YOLOv2 [31] predicts the BB as a 5-dimensional vector in a cell according to the anchor box, constraining the BB to not deviate from the grid cell, and if the original YOLO predicted the center of the grid, v2 predicts how far it moves from the top left vertex.

In this paper, the Wi-Fi data points represented by BBs are represented by one grid cell containing all of them, and the BBs are divided into grid cells containing the minimum number of BBs to represent the clusters of BBs and calculate the correlation of each grid cell. This grid cell is called a cell in this paper.

1.2. Contribution

The contributions of this study can be summarized as follows:

- The proposed technique improves the problem of specifying parameters for threshold setting that we encountered in our previous work;
- The clustering accuracy is improved by using our proposed grid cells instead of clustering data points with K-means clustering;
- The proposed learning model uses only Wi-Fi fingerprinting information to allow the RSSI of the previous location to affect the localization of the current location;
- The proposed technique enhanced the algorithm and movement path generation structure of the previous paper, and its performance was validated through experiments.

This paper is organized as follows: Section 2 describes the proposed technology, which visualizes all data points as BB used for image object detection, groups and classifies BBs into grid cells based on the results obtained by combining and expanding BBs and calculates the association information between grid cells to generate movement path data. Section 3 describes the performance evaluation of the proposed localization algorithm. Finally, Section 4 presents the conclusions and future work of this study.

2. Suggested Techniques

The technology introduced in this paper begins by generating associative information between non-contiguous data points. The process of representing all data points as BB, calculating the IoU between BBs, combining nested BBs, and scaling the size of the BBs is repeated until the target BB size is reached. And they are represented as cells to cluster the extended BBs. In this process, all BBs are designated as a single cell, and a cell division process is carried out to generate visualization information that represents the clustering of adjacent BBs. For movement path generation, the continuity of neighboring cells is calculated to generate adjacency information to determine whether a path can be created between cells. Finally, the movement path is generated utilizing the adjacency information of the cells. This process enhances the movement path generation algorithm from our previous paper to create the entire path. Thus, the proposed technique improves on the problems associated with clustering in our prior work and provides a basis for stable path computation. The following describes each step of the proposed technique in detail.

2.1. Bounding Box

We used BB to visualize and represent the Wi-Fi fingerprint data used for the experiment as image information based on X, Y coordinates. This process is divided into initial BB setup and BB joining, and the BB joining process consists of BB expansion and merge.

2.1.1. Initial Bounding Box Setup

All data points on the coordinate plane are replaced by BBs with a size of BBi from the initial fingerprint dataset D, as specified in Table 1. This creates a BB for all data points, and the coordinate values (X, Y) of the data points are the center of the BB. The initial data D is equal to Equation (2). In this formula, n is the number of RSSI signal values, N is the

number of APs, and k is the number of data contained in the dataset. Let $Y_{i,l}$ be the x (or l) coordinate of the i-th data point, and $Y_{i,m}$ be the y (or m) coordinate of the i-th data point.

$$x \in \mathbb{R}^n, -100 \leq x_i \leq 0 \; for \; all \; i = 1, \ldots, n, X = \{x_1, \ldots, x_k\}, Y = \begin{bmatrix} l_1, m_1 \\ l_2, m_2 \\ \cdots \\ l_k, m_k \end{bmatrix}, \; D = \{X, Y\} \tag{2}$$

Table 1. Hyperparameters required for merging BBs.

Hyper Parameter	BBi Size	m_factor	max_factor	IoU Threshold
value	0.5 m × 0.5 m	×1.5	5 m	0

2.1.2. Merging and Expanding Bounding Boxes

BB merging is a two-step process. First, the merge process combines two nested BBs to consolidate them. Secondly, there is an expansion process that increases the size of an independent BB that does not overlap with other BBs, allowing it to be merged with other BBs. The ratio variable for scaling BB in this process is m_factor, let us call it α. Hyperparameters are shown in Table 1. BB size is $W_{init} \times H_{init}$, which is the initial size of the BB. The maximum size of the BB, max_factor, is $S_{max} = (W_{max}, H_{max})$, the IoU Threshold is $\tau_{threshold}$, the information of each BB is $bb_j = (l_j, m_j, w_j, h_j)$, l is the x-axis position, m is the y-axis position, w is the width, h is the height, and D' is the dataset index included in the BB. Accordingly, BB information is represented as $BB = \{bb_1, \ldots, bb_t\}$. The initial BB setup is as shown in Equation (3) below.

$$\begin{aligned} BB = \{bb_1, \ldots, bb_k\}, forall bb_i \in BB, w(bb_i) = W_{init}, h(bb_i) \\ = H_{init}, l(bb_i) = y_{i,l} - \tfrac{W_{init}}{2}, m(bb_i) = y_{i,m} - \tfrac{H_{init}}{2} \end{aligned} \tag{3}$$

Next, calculate the IoU value between the input BBs using Equation (4).

$$getIoU(BB) = \begin{bmatrix} -1 & \tau_{1,2} & \cdots & \tau_{1,k} \\ \tau_{2,1} & -1 & \cdots & \tau_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \tau_{k,1} & \tau_{k,2} & \cdots & -1 \end{bmatrix} \tag{4}$$

To save the status information of the merging process, we set up a BB List to store and manage the BBs created in the current stage, and checked whether the merging or expanding process was performed. At this point, if any merging or expanding process is performed in the current step, it will proceed to the next step. If all BBs are larger than max_factor, the process will terminate immediately without proceeding with the merging and expanding process of the next step. Therefore, if the size of all BBs does not yet meet the termination condition, the process of combining and enlarging BBs will be repeated for the number of initial BBs. In this iteration, the IoU value of the current BB and other BBs are calculated and the BBs that are greater than the IoU Threshold value are merged. This checks to see if the current BB is already a combined BB, and at every iteration it checks for combining or expanding information per BB. It then calculates the minimum BB size of the current BB and the other selected BB to calculate the minimum BB that contains both BBs, creating a new combined BB and adding it to the BB List. The BB List is then used to update the status information of the BBs used in the union, specifying that they are the BBs used in the union. If BBs have an IoU value that is less than the IoU Threshold required for joining, they do not have overlapping regions and can be judged to be independently located BBs. Thus, independent BBs expand their size to merge with other BBs around them. It sets the expanded BB information to the BB List and updates the status of the

current BB. The expansion of a BB is conducted by multiplying the current size of the BB by m_factor. In our previous paper, we set a threshold in the form of a constant to prove neighboring data points [12]. If this is applied to the BB of this paper, the proximity of independent BBs can be determined using GIoU. However, when using GIoU, the same issue as using a threshold constant value arises. We solved this problem by extending the nested BBs using IoU. BBs that were not used for merging and expansion are processed in the next step. Checks for BBs that have not been merged or expanded in the current staff are saved to the BB List. Finally, as a condition for ending the merge and expand phase, the BB List information is checked to determine whether the iteration should continue or end. The behavior for merging and expanding BBs is shown in Algorithm 1. The hyperparameter definitions used in this algorithm are tabulated in Section 4.

Algorithm 1 Combination bounding box Algorithm

Input: Bounding Box set $BB = \{bb_1, \dots, bb_t\}$,
m_factor α,
IoU Threshold $\tau_{threshold}$,
max_factor $S_{max} = (W_{max}, H_{max})$
Output: updated Bounding Box set $BB^* = \{bb_1, \dots, bb_{t^*}\}$,
doRepeat

doRepeat ← *False*;
IoU ← getIoU(*BB*);
$BBSizeCheck \leftarrow \{i | BB_{i,w} < W_{max} \textbf{and} BB_{i,h} < H_{max}, i = 1,2, \dots, t\}$;
$mergedBBIdx \leftarrow \{\}$;
Initialize BB^*;
for *iter* ← 1 **to** $|BB|$ **do**
  if $BB_{iter,w} < W_{max}$ ***and*** $BB_{iter,h} < H_{max}$ **then**
    doRepeat ← *True*;
    $I = \{i | IoU_{iter,i} > \tau_{threshold}, i = 1,2, \dots, t\}$;
    $BBMergeCheck \leftarrow BBSizeCheck \cap I$;
    if $iter \notin mergedBBIdx$ ***and*** $0 < |BBMergeCheck|$ **then**
      $BBMergeCheck_{iter} \leftarrow True$;
      $mergedBBIdx \leftarrow mergedBBIdx \cup BBMergeCheck$;
      $l_{new} \leftarrow min\{BB_{i,l} | BBMergeCheck_i = True, i = 1,2, \dots, t\}$;
      $m_{new} \leftarrow min\{BB_{i,m} | BBMergeCheck_i = True, i = 1,2, \dots, t\}$;
      $w_{new} \leftarrow max\{BB_{i,l} + BB_{i,w} | BBMergeCheck_i = True, i = 1,2, \dots, t\}$;
      $h_{new} \leftarrow max\{BB_{i,m} + BB_{i,h} | BBMergeCheck_i = True, i = 1,2, \dots, t\}$;
      $D'_{new} \leftarrow \{D'_i | BBMergeCheck_i = True, i = 1,2, \dots, t\}$;
      $BB^* \leftarrow BB^* \cup \{(l_{new}, m_{new}, w_{new} - l_{new}, h_{new} - m_{new}, D'_{new})\}$;
    else if $|BBMergeCheck| < 0$ **then**
      $mergedBBIdx \leftarrow mergedBBIdx \cup \{iter\}$;
      $w_{new} \leftarrow \alpha BB_{iter,w}$;
      $h_{new} \leftarrow \alpha BB_{iter,h}$;
      $l_{new} \leftarrow BB_{iter,l} + \frac{BB_{iter,w} - w_{new}}{2}$;
      $m_{new} \leftarrow BB_{iter,m} + \frac{BB_{iter,h} - h_{new}}{2}$;
      $BB^* \leftarrow BB^* \cup \{(l_{new}, m_{new}, w_{new}, h_{new}, D'_{new})\}$;
    end
  end
  $BB^* \leftarrow BB^* \cup \{BB_i | i \notin BBSizeCheck, i = 1,2, \dots, t\}$;
End

2.2. Cell Division

After the BBs are combined and extended, a grouping operation is required to show their proximity to each other in a clustered form. To do this, neighboring BBs are clustered into cells, and each cell lays the foundation for the generation of path information. YOLO's grid cell refers to a box that predefines the shape and size of the predicted BBs. We initially represented the entire coordinate plane of the data points as a single cell, and then divide the cell into sub-cells for clustering the BBs. In other words, we used cell for clustering BBs.

Let $C = \{c_1, \ldots, c_r\}$ be a cell, and for each state j, denote the cell information by $c_j = (l_j, m_j, w_j, h_j)$. The minimum size of the cell is called min_factor and is represented by $S_{min} = (W_{min}, H_{min})$. Let W_{min} be the minimum width and H_{min} the minimum height of the cell. The variable for limiting the number of BBs per direction within a cell is max_overlap_factor, letting β be the number of BBs. This allows the initialization of the first cell to be $C = \{c_1\}$ and c1 to be equal to Equation (5).

$$c_1 = (min(Y_{:,l}), min(Y_{:,m}), max(Y_{:,l}) - min(Y_{:,l}), max(Y_{:,m}) - min(Y_{:,m}), BB) \quad (5)$$

Cells are used to cluster BBs because BBs have overlapping regions in neighboring cells or the IoU value between BBs is 0. In other words, if BBs are independent and not overlapping, it is difficult to generate adjacency information between BBs based on BBs alone. Therefore, we group BBs into cells and determine if they can be moved to a neighboring cell. At this time, we determined where the BB is located inside the cell (top, bottom, left, right, etc.) and expressed the adjacency to create a movement path between cells. Therefore, cells are divided based on BBs, and as a result, in regions where the initial data points are not densely located, the cell size becomes relatively large. In this case, the problem of assigning adjacency to non-adjacent BBs and generating incorrect movement paths has been resolved.

The construction of a cell below describes the process of setting up an initial cell to represent all data points represented by BBs as a single cell, and then iteratively dividing the entire cell into sub-cells consisting of adjacent BBs.

2.2.1. Initial Cell Settings

We display all the BBs that have been merged and expanded as images on the coordinate plane and set up a single cell of a size that encompasses the entire BB. Unlike the checkerboard cell setup typically used by cell in YOLO, we start with a single cell and iterate through the cell, dividing it based on the BBs clustered inside the cell.

2.2.2. Dividing into Sub-Cells

The input values for dividing cells are the current cell size and current BB information, followed by the minimum size of the divided cell, the number of BBs the cell contains, and finally the data points. The result of the cell partitioning process is the partitioned cell information table C. Basically, one cell is divided into 4 equal parts, and the size c_j of the divided cell is compared with S_{min}, which represents the min_factor, to check. In this case, two conditions arise: firstly, the division proceeds if the size c_j of the divided cell is greater than S_{min}, and secondly, the division stops if the size c_j of the divided cell is smaller than S_{min} and the division condition is not met. The division process for each condition is as follows:

1. Dividing process: First, perform a dividing condition discovery process. Temporarily divide the cell into four parts based on the center point of the current cell. The number of BBs included in the temporarily divided area is calculated, and if the count is greater than β, which is the max_overlap_factor, the division is confirmed. At this point, the divided cell information is recorded in the table, and the division process is repeated based on the divided cell. If the number of BBs in each divided region is less than β, the division process is terminated;

2. End of division: Records the size of the current cell where the division ended, and the BB information contained in the current cell in the declared cell information table. Check where in the cell the BB contained in the current cell is located. This checks the location of nine regions inside the current cell, categorized as top, bottom, left, right, and each corner. Check the BBs contained in the categorized location and store and update that information in the cell information table.

The behavior of performing cell segmentation of the combined BB based on all data points is shown in Algorithm 2. The hyperparameters used in this algorithm are defined in a table in Section 4.

Algorithm 2 cell Segmentation Algorithm

Input: cell infomation c_{input}
Bounding Box set $BB = \{bb_1, \dots, bb_t\}$,
max_overlap_factor β,
min_factor $S_{min} = (W_{min}, H_{min})$,
Output: cell set $C^* = \{c_1, \dots, c_{r^*}\}$,
doDivision ← *False;*
IoU ← getIoU*(BB);*
$BBSizeCheck \leftarrow \{i | BB_{i,w} < W_{max}$ ***and*** $BB_{i,h} < H_{max}, i = 1,2, \dots, t\}$;
Initialize C^*;

if $W_{max} < \frac{BB_{iter,w}}{2}$ ***and*** $H_{min} < \frac{BB_{iter,h}}{2}$ **then**
$halfW \leftarrow \frac{c_{input,w}}{2}$;
$halfH \leftarrow \frac{c_{input,h}}{2}$;
$leftL \leftarrow c_{input,l}$;
$rightL \leftarrow c_{input,l} + \frac{c_{input,w}}{2}$;
$topM \leftarrow c_{input,h} + \frac{c_{input,h}}{2}$;
$bottomM \leftarrow c_{input,h}$;
$C' \leftarrow \begin{Bmatrix} (leftL, topM, c_{input,w}, halfH), (leftL, bottomM, c_{input,w}, halfH), (rightL, bottomM, halfW, c_{input,h}), \\ (rightL, bottomM, halfW, c_{input,h}) \end{Bmatrix}$;
$BBOverlapNum \leftarrow \{0,0,0,0\}$;
for *iter* ← 1 **to** $|C'|$ **do**
if $\left| \left\{ i \middle| \begin{matrix} C'_{iter,l} \le BB_{I,l} + BB_{I,w} \textbf{ and } C'_{iter,l} + C'_{iter,w} \ge BB_{I,l} \textbf{ and } C'_{iter,m} \le BB_{I,m} + BB_{I,h} \\ \textbf{and } C'_{iter,m} + C'_{iter,l} \ge BB_{I,m}, i = 1,2, \dots, k \end{matrix} \right\} \right| > \beta$ **then**
doDivision ← *True;*
break;
end
end
end
if *doDivision =True* **then**
$C' = \begin{Bmatrix} \left(c_{input,l}, c_{input,m}, \frac{c_{input,w}}{2}, \frac{c_{input,h}}{2}\right), \left(c_{input,l} + \frac{c_{input,w}}{2}, c_{input,m}, \frac{c_{input,w}}{2}, \frac{c_{input,h}}{2}\right), \\ \left(c_{input,l}, c_{input,m} + \frac{c_{input,h}}{2}, \frac{c_{input,w}}{2}, \frac{c_{input,h}}{2}\right), \left(c_{input,l} + \frac{c_{input,w}}{2}, c_{input,m} + \frac{c_{input,h}}{2}, \frac{c_{input,w}}{2}, \frac{c_{input,h}}{2}\right) \end{Bmatrix}$;
for *iter* ← 1 **to** $|C'|$ **do**
$C^* \leftarrow C^* \cup \{Algorithm2(C'_{iter}, BB, \beta, S_{min})\}$;
end
else
$C^* \leftarrow C^* \cup \{c_{input}\}$;
end

2.3. Adjacency Check

The adjacency check determines whether a route can be established to a neighboring cell based on the positions of BBs inside the cell. This means that even if cells are adjacent to each other, a route should not be created if the BBs inside the cell are not adjacent. In this case, we need the direction in which the cell is activated to find the cell's adjacency. To do this, we define the directionality as $direction(c_i, BB) \in \mathbb{R}^8$. This is a function that returns the position of the BB contained in the i-th cell region in 8 directions, as determined according to Algorithm 3.

Algorithm 3 cell direction Algorithm

Input: cell information c_{input}
Bounding Box set $BB = \{bb_1, \ldots, bb_t\}$,
Output: cell direction $direction \in \mathbb{R}^8$

/* The order of direction information is up, down, left, right,
top left, bottom right, bottom left, top right. */
$direction \leftarrow (False, False, False, False, False, False, False, False)$;
$heightOneThird \leftarrow \frac{c_{input,h}}{3}$;
$widthOneThird \leftarrow \frac{c_{input,w}}{3}$;

$$c' \leftarrow \left\{ \begin{array}{c} \left(c_{input,l}, c_{input,r} + heightOneThird \times 2, c_{input,w}, heightOneThird\right), \\ \left(c_{input,l}, c_{input,r} + heightOneThird, c_{input,w}, heightOneThird\right), \\ \left(c_{input,l}, c_{input,r}, widthOneThird, c_{input,h}\right), \left(c_{input,l}, c_{input,r} + widthOneThird \times 2, widthOneThird, c_{input,h}\right) \end{array} \right\};$$

for $i \leftarrow 1$ **to** $|c'|$ **do**
  for j $\leftarrow 1$ **to** $|BB|$ **do**
    $IoU \leftarrow getIoU(\{c_i'\} \cup BB)$;
    if $\max(IoU_{1,:}) > 0$ **then**
      $direction_i = True$;
      break;
    end
  end
end
$direction_5 = direction_1 \cap direction_3$; /* top left */
$direction_6 = direction_2 \cap direction_4$; /* bottom right */
$direction_7 = direction_2 \cap direction_3$; /* bottom left */
$direction_8 = direction_1 \cap direction_4$; /* top right */

2.3.1. Calculating Adjacency

Adjacency generation defines the space for movement path generation as an adjacency matrix A, where each element is either 0 or 1, as shown in Equation (6).

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,r} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r,1} & a_{r,2} & \cdots & a_{r,r} \end{bmatrix}, a \in \{0,1\}, a_{i,i} = \begin{cases} 1, \text{True} \in direction(c_i, BB) \\ 0, otherwise \end{cases} \quad (6)$$

where $\alpha_{i,j}$ represents the (i, j)-th element, and the diagonal element $(i = j)$ has a value if and only if BB exists.

The calculation order is iterated over the number of cells in the adjacency matrix, and if a cell contains a BB, it is checked whether the current cell and its neighboring cell are adjacent to each other. Algorithm 4 demonstrates the process of checking if there is a BB

inside the current cell and its neighboring cells that matches the adjacent direction. If the condition is met, it is registered in the adjacency list.

Algorithm 4 cell Adjacency Algorithm

Input: cell information set $C = \{c_1, \ldots, c_r\}$
Bounding Box set $BB = \{bb_1, \ldots, bb_t\}$
Output: cell adjacency matrix $A \in \mathbb{R}^{r \times r}$
$directions \leftarrow \{\}$;
Initialize A;
for $i \leftarrow 1$ **to** $|C|$ **do**
  $directions \leftarrow directions \cup \{direction(c_i, BB)\}$;/*algorithm 3*/
end
for $i \leftarrow 1$ **to** $|C|$ **do**
  if $True \in directions_i$ **then**
    $A_{i,i} \leftarrow 1$;
    $left_i \leftarrow c_{i,l}$;
    $right_i \leftarrow c_{i,l} + c_{i,w}$;
    $top_i \leftarrow c_{i,m} + c_{i,h}$;
    $bottom_i \leftarrow c_{i,m}$;

    for $j \leftarrow i + 1$ **to** $|C|$ **do**
      $left_j \leftarrow c_{j,l}$;
      $right_j \leftarrow c_{j,l} + c_{j,w}$;
      $top_j \leftarrow c_{j,m} + c_{j,h}$;
      $bottom_j \leftarrow c_{j,m}$;
      $direction_idx \leftarrow 0$;
      if $(top_i = bottom_j)$ **and** $\left((left_i < right_j and left_i \geq left_j)\ \mathbf{or}\ (right_i \leq right_j and right_i > left_j)\right)$ **then**
        $direction_idx \leftarrow 1$;
      else if $(top_j = bottom_i)$ **and** $\left((left_j < right_i and left_j \geq left_i)\ \mathbf{or}\ (right_j \leq right_i and right_j > left_i)\right)$ **then**
        $direction_idx \leftarrow 2$;
      else if $(right_j = left_i)$ **and** $\left((bottom_j < top_i and bottom_j \geq bottom_i)\ \mathbf{or}\ (top_j \leq top_i and top_j > bottom_i)\right)$ **then**
        $direction_idx \leftarrow 3$;
      else if $(right_i = left_j)$ **and** $\left((bottom_i < top_j and bottom_i \geq bottom_j)\ \mathbf{or}\ (top_i \leq top_j and top_i > bottom_j)\right)$ **then**
        $direction_idx \leftarrow 4$;
      else if $(top_i = bottom_j)$ **and** $(left_i = right_j)$ **then**
        $direction_idx \leftarrow 5$;
      else if $(bottom_i = top_j)$ **and** $(right_i = left_j)$ **then**
        $direction_idx \leftarrow 6$;
      else if $(bottom_i = top_j)$ **and** $(left_i = right_j)$ **then**
        $direction_idx \leftarrow 7$;
      else if $(top_i = bottom_j)$ **and** $(right_i = left_j)$ **then**
        $direction_idx \leftarrow 8$;
      end
      if $direction_{idx} > 0$ **then**
        $has_direction_i \leftarrow directions_{i,direction_{idx}}$;
        if $mod(direction_{idx}, 2)$**then**
          $has_direction_j \leftarrow directions_{j,direction_{idx}+1}$;
        else
          $has_direction_j \leftarrow directions_{j,direction_{idx}-1}$;
        end
        if $has_direction_i = has_direction_j$ **then**
          $A_{i,j} \leftarrow 1$;
          $A_{j,i} \leftarrow 1$;
        end
      end
    end
  end
end

2.3.2. Movement Path Generation

The process of generating movement paths is iterated for each training data point. The process begins by selecting a random cell and sampling the data points within that cell. This is followed by a random walk in the neighbor matrix of the current cell. A random walk samples the data points contained in a cell at a specific location. If this cell is the last one in the path to be generated, the corresponding coordinate values (x, y) are used as the label. The algorithm for this is a refinement of Algorithm 3 from our previous paper [12], expressed as Algorithm 5. Algorithm 5 is a movement path data generation algorithm. It takes a dataset D, the state transition matrix P from the previous paper, the length of the movement path, cell information, and cell adjacency matrix A as inputs. The algorithm produces an output dataset defined as D'. It collects the data points x contained within cells and uses the cell number of the last point. Separate datasets are created for training and testing purposes. Accordingly, the output data D' and each element t are defined by Equation (7).

$$D' = \{t_1, t_2, \ldots, t_{k'}\}, t_i = \left(\left(x_1, x_2, \ldots, x_{path}\right), y\right), y \in \{1, 2, \ldots, r\} \quad (7)$$

Algorithm 5 Movement Path Generation Algorithm [12]

Input: dataset $D = \{X, Y\}$,
state transition matrix P,
Path length *path*,
number of path to create k',
cell information set $C = \{c_1, \ldots, c_r\}$,
cell adjacency matrix A
Output: sequence data D'
$D' \leftarrow \{\}$;
for *iter* $\leftarrow$ *1* **to** k' **do**
$cellidx \leftarrow idx \in \{i | a_{i,j} = 1, i = 1,2, \ldots, r\}$;
$x' \leftarrow \{\}$;
$x' \leftarrow x' \cup \{x \in \{x_i | c_{cellidx,l} \leq Y_{i,l} \wedge Y_{i,l} \leq c_{cellidx,l} + c_{cell_{idx},w} \wedge c_{cellidx,m} \leq Y_{i,l} \wedge Y_{i,l} \leq c_{cellidx,m} + c_{cell_{idx},h}, i = 1,2, \ldots, k\}\}$;
for *pathiter* $\leftarrow$ *2* **to** path **do**
$cellidx \leftarrow idx \in \{i | a_{i,j} = 1, i = 1,2, \ldots, r\}$;
$x' \leftarrow x' \cup \{x \in \{x_i | c_{cellidx,l} \leq Y_{i,l} \wedge Y_{i,l} \leq c_{cellidx,l} + c_{cell_{idx},w} \wedge c_{cellidx,m} \leq Y_{i,l} \wedge Y_{i,l} \leq c_{cellidx,m} + c_{cell_{idx},h}, i = 1,2, \ldots, k\}\}$;
end
$D' \leftarrow D' \cup \{(x', cellidx)\}$;
End

3. Experiment

3.1. Experimental Environment

We analyze the performance of the proposed techniques using publicly available Wi-Fi fingerprint datasets. The dataset used is from a 2019 paper [24].

The dataset used in this paper contains RSSI, latitude, longitude, and layer information. Specifically, latitude and longitude data were calculated as datapoint locations. The training dataset includes 3852 data from the 0th floor and 3323 data from the 1st floor, with a total of 489 indoor positioning points used. The dataset also contains 443 unique APs. However, in this paper, we only used data from floor 0 because we do not consider interfloor movement. Considering only layer 0 data, the number of unique APs was found to be 173. The following hyperparameters are used to calculate the movement paths according to the proposed method.

Table 1 lists the hyperparameters required for merging BBs. Here, BBi size refers to the size of the initial BB, m_factor represents the multiplier used for expanding the BB when there are no neighboring BBs around it, and max_factor is the maximum size for combining and expanding the BB, which cannot exceed 5 m. The IoU Threshold is 0. Non-overlapping BBs cannot be merged with neighboring BBs, only expanded.

Table 2 presents the hyperparameters required to divide cells, where the minimum size of the divided cell is 5 m and the maximum number of BBs contained in the cell is limited to 2. If this limit is exceeded, the cell is divided.

Table 2. Hyperparameters for Dividing cells.

Hyper Parameter	min_factor	max_overlap_factor
Value	5 m	2

The information of the dataset needed for training is shown in Table 3, where the units are counts. Use Algorithm 5 to visit cells along the movement path from a random cell and collect RSSI data for each cell. The RSSI data from the visited cell is gathered by splitting it into train and test datasets. The last visited cell is numbered Y. Based on this data, we trained a model for the classification problem, utilizing a single slot on an Nvidia DGX A100 as the training environment.

Table 3. Hyperparameters for the dataset.

Hyper Parameter	Data Point	APs	Path	Training Data	Test Data
Value	3852	173	5	30,000	5000

Table 4 displays the range of hyperparameters we employed for training, subsequently serving as the search space for the Bayesian optimizer. The Bayesian optimizer is a method that predicts the optimal values of hyperparameters based on a prior probability distribution, which was used to find the optimal hyperparameters. The optimal hyperparameters calculated from the experiments are summarized in Table 5.

Table 4. Range of hyperparameters.

Hyper Parameter	Learning Rate	Dropout	LSTM Length	Hidden Cell	Epoch	RSSI Normalize	Batch Size
value	0.00001~0.01	0.0~0.4	[2,3,4,5,6,7]	[64,128,256,512,1024]	50~5000	[True, False]	[64,128,256,512,1024]

Table 5. Best Model in Learning.

Hyper Parameter	Learning Rate	Dropout	LSTM Length	Hidden Cell	Epoch	RSSI Normalize	Batch Size
value	0.000676	0.188579	3	64	3552	TRUE	1024

The initial data distribution before training the LSTM on the Wi-Fi fingerprint data used in the experiment is shown in Figure 2. This shows the distribution of data based on the Wi-Fi signal alone, regardless of the indoor structure.

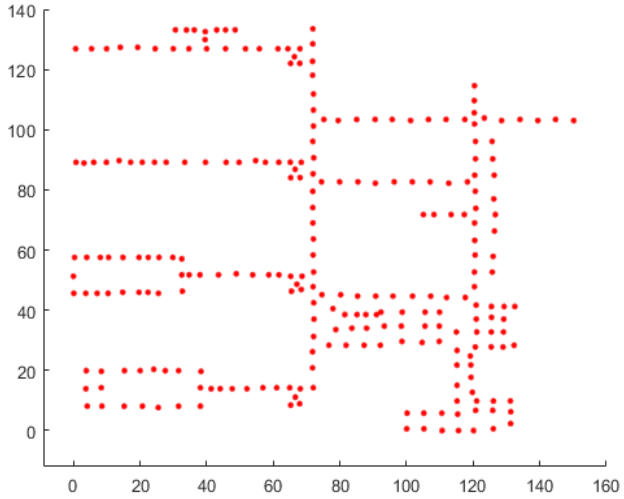


Figure 2. Distributions of initial data points.

3.2. Experimental Results

Based on the RSSI signal data, the proposed technique is applied to represent the BB, and the final BB is obtained through the process of merging and expanding the BB. The result is then divided into cells, as shown in (a) of Figure 3. The initial cell is sized to encompass all BBs placed within the floor, and the distribution of BBs across the cells demonstrates that the BBs and cells form the foundation for generating movement paths. This is (b) in Figure 3. As a result, we can observe that the BBs positioned within the entire cell represent the gathered dataset as BBs, and they are organized and divided in a manner consistent with the actual indoor structure of the floor.

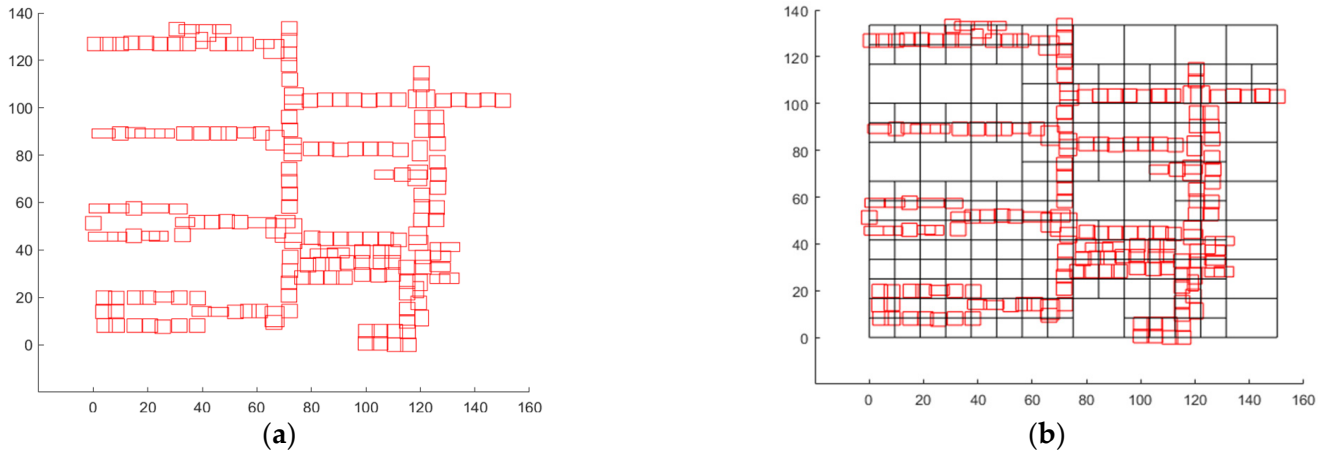


Figure 3. BB and cell division. (**a**) The result of merging and expanding the BB. (**b**) The result of merging the BB from (**a**) with the structure of the cells divided by the cell division algorithm.

When the placement of these BBs is plotted against the arrangement of the original data points, we can observe that they are correctly distributed within the cell region. This proves the appropriateness of the cell division, as the clustering of the replaced BBs is correctly represented within the cells. Therefore, our cell division technique is the basis for computing the movement path. In Figure 4a shows that the division of the cells made sense when merged with the original data points.

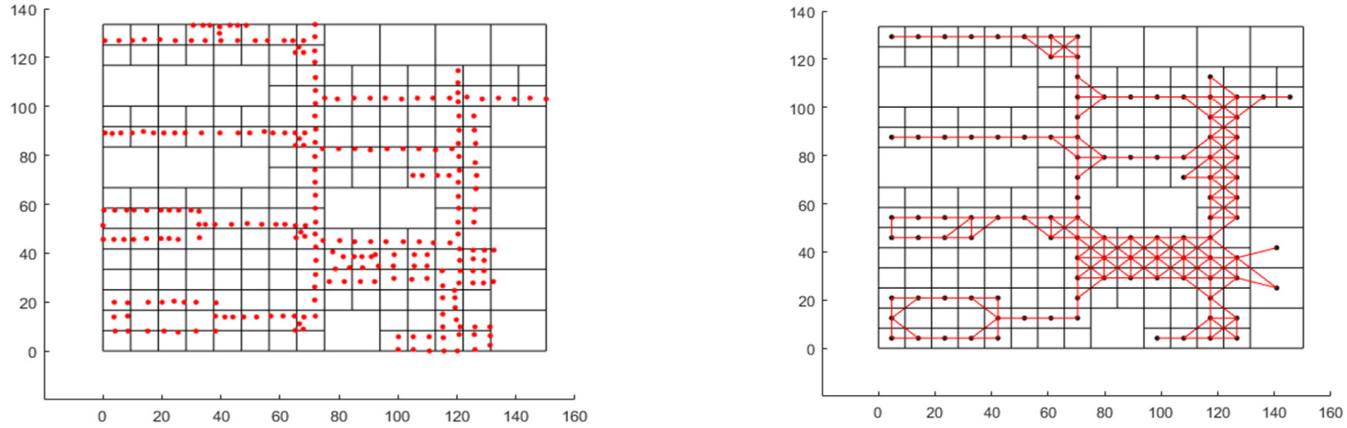


Figure 4. Cell division and movement path computation: (**a**) Distribution the initial data with the cell structure divided by the cell algorithm. (**b**) Paths generated by the movement path generation algorithm.

Based on the BB's location within the cell calculated in (b) of Figure 3 and the neighboring cells where the BB exists, the cell adjacency calculation algorithm determines the adjacency. The movement path was then computed based on cells whose adjacency was confirmed through the path generation algorithm. The result is depicted in (b) of Figure 4, and we confirmed that the calculated paths match the actual paths taken in the real building.

Training was carried out by utilizing the computed movement path dataset and the specified hyperparameters. The results are shown in Table 6, which are the experimental results of the proposed method based on the 2019 dataset [32]. The experimental results express the accuracy of the computed paths. The training data and test data were trained with the LSTM model, and the accuracy of the training results is shown as the mean, deviation, maximum, and minimum values. We can see that the difference between the maximum and minimum accuracy is 0.5% for the test data, and the deviation is low, so the accuracy remains constant.

Table 6. Experimental results (unit: Accuracy %).

Model	Average	Deviation	Max	Min
LSTM (path = 5)	93.968	0.327	94.48	93.64

An experiment was conducted to determine the optimal path length for the proposed technology. When experiments were conducted on various paths using hyperparameter search, the performance of the top five results was selected and evaluated. Table 7 displays the results of an experiment comparing the performance when applying different numbers of paths (accumulated past data). The number of paths used in the experiment was confirmed to be 1 case without a path, 3, 5, and 7 using paths. In experiments using 1 to 5 paths, the accuracy of location judgment gradually improved, while with 7 paths, the accuracy decreased. This confirmed that the optimal number of paths for location judgment is 5.

Table 7. Experiment with different path counts for the proposed technology (unit: Accuracy %).

Number of Path	Path 1	Path 3	Path 5	Path 7
Average	93.548	93.408	93.968	92.894
Deviation	0.2051	0.3417	0.327	0.3801
Max	93.74	93.8	94.48	93.5
Min	93.24	93	93.64	92.53

We conducted an experiment to determine the optimal path length for the proposed technology. When we performed experiments on various paths using hyperparameter search, we selected and evaluated the performance of the top five results. Table 7 shows the results of an experiment comparing the performance when applying different numbers of paths (accumulated past data). The number of paths used in the experiment was 1 case without a path, and 3, 5, and 7 cases with paths. In experiments using 1 to 5 paths, the accuracy of location estimation gradually improved, while with 7 paths, the accuracy decreased. This confirmed that the optimal number of paths for location estimation is 5.

To validate the performance of the clustering method using the proposed cells in this study, we compared it with K-means clustering. The validation experiment involved clustering areas based on the same dataset and training a model to estimate the location of each area. We set up the same task for further comparison. The number of clusters was determined to be 150 through experiments, in order to create a K-means clustering area of a size similar to the minimum size of the grid cell proposed by our approach, which was then applied in the comparative experiment. To compare the performance of this experiment, we used KNN and FNN models. The comparison results are presented in Table 8, and it was confirmed that the grid cell-based clustering method proposed by our approach exhibited superior accuracy and performance improvements across all models when compared to the K-means clustering method.

Table 8. Comparison of Clustering Techniques (unit: Accuracy %).

Model	Technique Comparison	K-Means Clustering (Cluster Count 150)	Grid Cell-Based Clustering
KNN	Find (k = 1)	91.8	92.8
	Medium (k = 10)	68.9	92.5
	Coarse (k = 100)	26.5	71.8
FNN	Layer = 1	88.2	91.2
	Layer = 2	79.2	86.7
	Layer = 3	75.9	79.3

There is a need to conduct experiments with other models to validate the performance of the proposed technology. However, to utilize continuous time-series information such as path data, LSTM must be employed. As a result, the comparison subject was divided into cases where route information is utilized and cases where it is not. KNN and FNN were utilized as the latter models. In the experiment, grid cells were applied to the proposed technology, and its performance was verified in an experiment with one path, where path information was not used for comparison with other models. Additionally, it was tested in an environment with five paths to confirm the maximum performance of the proposed technology. The accuracy of the experimental results was compared with the performance of the models using a classification problem that predicts the cell to which signal data belongs. When there are five paths, learning was conducted and evaluated as a classification problem, predicting the cell to which the final signal data belongs. The results are displayed in Table 9, confirming that the performance of the LSTM model for the proposed technology surpassed that of other models.

Table 9. Performance comparison of the proposed technique with other models (unit: Accuracy %).

Model	Path	Technique Comparison	Accuracy
KNN	1	Find (k = 1)	92.8
		Medium (k = 10)	92.5
		Coarse (k = 100)	71.8
KNN	1	Layer = 1	91.2
		Layer = 2	86.7
		Layer = 3	79.3
LSTM	n	Path = 1	93.5
		Path = 5	94

In the experiments for this paper, we used a minimum of 5 m as the threshold in the cell division size problem, because clustering could not be performed with a smaller value. Therefore, the bounding box expansion size was limited to a maximum of 5 m. Moreover, there is a limitation that accumulated Wi-Fi RSSI is required for accurate location estimation.

4. Conclusions

In this paper, we proposed a method to generate movement path data based on data without continuity information of signals collected by Wi-Fi RSSI. We confirmed the applicability of the proposed technology as a technique to reduce the positioning error of indoor location-based services. We proposed a technique for generating movement path data by representing the data points of Wi-Fi fingerprint data as Bounding Boxes, merging and expanding BBs using IoU, and forming clusters of data points based on grid cells. Wi-Fi fingerprint data for indoor environments lack sequence information. Therefore, the input data for learning should consist of continuous datasets rather than isolated RSSI values. To solve this problem, this paper constructs BBs from indoor positioning points of Wi-Fi fingerprint data. Based on this, grid cells are created to divide the dataset area. Adjacency is calculated based on the BBs distributed within each cell, enabling the creation of movement paths. In the previous paper, the cluster size was not constant, the number of clusters was fixed, the shape of the cluster was not standardized, and data points that should not be included in the cluster were included. As a result, a problem was identified where the movement path was inaccurately generated due to clustering errors. In this paper, we have proposed improvements to address these issues.

The experimental results utilizing the proposed technique demonstrate that the cluster size is a minimum of 5 m. Consequently, the number of clusters can be adjusted flexibly based on the data point distribution. The clustering process utilizes a cell structure to prevent the inclusion of incorrect data points. As a result, it was verified that the movement path exhibited accuracy and a standardized cluster structure. Through experimentation, it was confirmed that an accurate movement path was established within an indoor positioning environment that necessitates time-series information. In addition, we were able to enhance the reliability of the adjacency calculation by addressing the issue of establishing connections between unrelated data points, which arose from the use of an arbitrary threshold in calculating data point adjacencies. And as a result of verifying the similarity between the Wi-Fi fingerprint data and the building structure from which it was collected, we were able to cluster the data points accurately compared to the unsupervised learning clustering method.

In the future, research needs to be expanded to include three-dimensional movement path calculations using a voxel structure for generating paths within a three-dimensional environment.

Author Contributions: Conceptualization, H.-G.S., D.L., C.-G.H. and C.-P.Y.; methodology, C.-G.H. and C.-P.Y.; software, H.-G.S. and C.-P.Y.; validation, H.-G.S., C.-G.H. and C.-P.Y.; formal analysis, C.-G.H. and C.-P.Y.; investigation, D.L. and C.-G.H.; resources, H.-G.S.; data curation, D.L. and C.-G.H.; writing—original draft preparation, H.-G.S. and C.-P.Y.; writing—review and editing, C.-G.H.

and C.-P.Y.; visualization, H.-G.S., D.L., C.-G.H. and C.-P.Y.; supervision, C.-G.H. and C.-P.Y.; project administration, C.-G.H. and C.-P.Y.; funding acquisition, C.-P.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Feng, C.; Au, W.S.A.; Valaee, S.; Tan, Z. Received-signal-strength-based indoor positioning using compressive sensing. *IEEE Trans. Mob. Comput.* **2011**, *11*, 1983–1993. [CrossRef]
2. Sahar, A.; Han, D. An LSTM-based indoor positioning method using Wi-Fi signals. In Proceedings of the ACM International Conference on Vision, Image and Signal Processing (ICVISP), Las Vegas, NV, USA, 27–29 August 2018.
3. Shrestha, S.; Talvitie, J.; Lohan, E.S. Deconvolution-based indoor localization with WLAN signals and unknown access point locations. In Proceedings of the International Conference on Localization and GNSS (ICL-GNSS), Turin, Italy, 25–27 June 2013.
4. Park, C.U.; Shin, H.-G.; Choi, Y.-H. A parallel artificial neural network learning scheme based on radio wave fingerprint for indoor localization. In Proceedings of the 10th International Conference on Ubiquitous and Future Networks (ICUFN), Prague, Czech Republic, 3–6 July 2018.
5. Tian, Z.; Tang, X.; Zhou, M.; Tan, Z. Fingerprint indoor positioning algorithm based on affinity propagation clustering. *EURASIP J. Wirel. Commun. Netw.* **2013**, *2013*, 272. [CrossRef]
6. Zhang, W.; Liu, K.; Zhang, W.; Zhang, Y.; Gu, J. Wi-Fi positioning based on deep learning. In Proceedings of the IEEE International Conference on Information and Automation (ICIA), Hailar, China, 28–30 July 2014; pp. 1176–1179.
7. Jarawan, T.; Kamsing, P.; Tortceka, P.; Manuthasna, S.; Hematulin, W.; Chooraks, T.; Phisannupawong, T.; Sanzkarak, S.; Munakhud, S.; Somjit, T. Wi-Fi Received Signal Strength-based Indoor Localization System Using K-Nearest Neighbors fingerprint integrated D* algorithm. In Proceedings of the 23rd International Conference on Advanced Communication Technology (ICACT), Pyeongchang, Republic of Korea, 7–10 February 2021; pp. 242–247.
8. Wang, G.; Abbasi, A.; Liu, H. Wi-Fi-based Environment Adaptive Positioning with Transferable Fingerprint Features. In Proceedings of the IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), Virtual Conference, 27–30 January 2021; pp. 123–128.
9. Shang, S.; Wang, L. Overview of WiFi Fingerprinting-based Indoor Positioning. *IET Commun.* **2022**, *16*, 725–733. [CrossRef]
10. Feng, X.; Nguyen, K.A.; Luo, Z.A. survey of deep learning approaches for WiFi-based indoor positioning. *J. Inf. Telecommun.* **2022**, *6*, 163–216. [CrossRef]
11. Wang, L.; Shang, S.; Wu, Z. Research on indoor 3D positioning algorithm based on wifi fingerprint. *Sensors* **2023**, *23*, 153. [CrossRef] [PubMed]
12. Shin, H.-G.; Choi, Y.-H.; Yoon, C.-P. Movement path data generation from Wi-Fi fingerprints for recurrent neural networks. *Sensors* **2021**, *21*, 2823. [CrossRef] [PubMed]
13. Zhao, Z.; Lou, Z.; Wang, R.; Li, Q.; Xu, X. I-WKNN: Fast-speed and high-accuracy WIFI positioning for intelligent sports stadiums. *Comput. Electr. Eng.* **2022**, *98*, 107619. [CrossRef]
14. Oh, A. Design of System for Accurate Tracking Services in Environments with Obstacles. *J. Inf. Commun. Converg. Eng.* **2021**, *19*, 161–165.
15. Gao, K.; Wang, H.; Nazarko, J.; Chobanov, G. Indoor Trajectory Prediction Algorithm Based on Communication Analysis of Built-In Sensors in Mobile Terminals. *IEEE Sens. J.* **2021**, *21*, 25234–25242. [CrossRef]
16. Wang, P.; Koike-Akino, T.; Orlik, P. Fingerprinting-Based Indoor Localization with Commercial MMWave Wi-Fi: NLOS Propagation. In Proceedings of the IEEE Global Communications Conference 2020 (IEEE Globecom 2020), Taipei, Taiwan, 7–11 December 2020.
17. Echizenya, K.; Kondo, K.; Kitagawa, T. Evaluation of the real-time indoor location and motion direction estimation system applying DNN to RSSI Fingerprints of BLE beacons. In Proceedings of the IEEE 9th Global Conference on Consumer Electronics (GCCE), Kobe, Japan, 13–16 October 2020.
18. Gu, Y.; Zhou, C.; Wieser, A.; Zhou, Z. Trajectory Estimation and Crowdsourced Radio Map Establishment from Foot-Mounted IMUs, Wi-Fi Fingerprints, and GPS Positions. *IEEE Sens. J.* **2019**, *19*, 1104–1113. [CrossRef]
19. Khassanov, Y.; Nurpeiissov, M.; Sarkytbayev, A.; Kuzdeuov, A.; Varol, H. Finer-level Sequential Wi-Fi-based Indoor Localization. In Proceedings of the 2021 IEEE/SICE International Symposium on System Integration (SII), Iwaki, Fukushima, Japan, 11–14 January 2021; pp. 163–169.
20. Li, Z.; Zhang, Y. Constrained ESKF for UAV Positioning in Indoor Corridor Environment Based on IMU and WiFi. *Sensors* **2022**, *22*, 391. [CrossRef] [PubMed]

21. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
22. Gers, F.A.; Schmidhuber, J.; Cummins, F. Learning to forget: Continual prediction with LSTM. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), Edinburgh, UK, 7–10 September 1999.
23. Mikolov, T.; Karafiat, M.; Burget, L.; Cernocky, J.; Khudanpur, S. Recurrent neural network-based language model. In Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH 2010), Chiba, Japan, 26–30 September 2010.
24. Van, H.G.; Mosquera, C.; Nápoles, G. A review on the long short-term memory model. *Artif. Intell. Rev.* **2020**, *53*, 5929–5955.
25. Rijayanti, R.; Jin, K.; Hwang, M. LSTM Model-based Prediction of the Variations in Load Power Data from Industrial Manufacturing Machines. *J. Inf. Commun. Converg. Eng.* **2022**, *20*, 295–302. [CrossRef]
26. Hwang, C.; Kim, H.; Jung, H. Detection and Correction Method of Erroneous Data Using Quantile Pattern and LSTM. *J. Inf. Commun. Converg. Eng.* **2018**, *16*, 242–247. [CrossRef]
27. Chen, Y.; Juan, T.; Jung, H. Text Classification on Social Network Platforms Based on Deep Learning Models. *J. Inf. Commun. Converg. Eng.* **2023**, *21*, 9–16. [CrossRef]
28. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 658–666.
29. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 12993–13000.
30. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
31. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
32. González, J.L.S.; Morillo, L.M.S.; Álvarez-García, J.A.; Ros, F.E.D.S.; Ruiz, A.R.J. Energy-efficient indoor localization Wi-Fi-fingerprint system: An experimental study. *IEEE Access* **2019**, *7*, 162664–162682. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.