A New Filtering Algorithm for Generating Path Loss Model to Improve Accuracy of Trilateration-Based Indoor Positioning System

L Susanto, W Wibisono, and T Ahmad

Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Email: lucassusanto@gmail.com, waswib@if.its.ac.id, tohari@if.its.ac.id

Abstract. Global Positioning System (GPS) is used to determine someone’s or something’s position. It is very accurate. But, GPS cannot be used in an indoor environment as its signal is blocked by an object (i.e. roof). To overcome this problem, Indoor Positioning System (IPS) is used in the indoor environment. There are two methods that IPS can use to determine the position. The first method is pattern matching. This method is highly accurate. But, the computation cost is expensive. The second method is the path loss model. Its computation cost is cheap. However, it is fairly accurate. In this paper, a new filter algorithm is proposed to improve path loss model accuracy. The algorithm works by filtering the received signal strength index (RSSI) in the training phase. The algorithm works by replacing the RSSI value at a certain distance by considering the current and the previous RSSI value. The experiment result shows that the proposed method has 3% better accuracy than the existing method.

1. Introduction

Industry revolution 4.0 makes many computer systems network connected. Computers become smarter and human works become increasingly automated. In this industrial revolution, the internet of things (IoT) has been increasingly demanded to automate human works.

Location-based service (LBS) is one of the applications of IoT. In LBS, the application is installed on a mobile device. So, its position changes when the device moves. One of the applications of LBS is an indoor tracking system [1]. It demands an accurate prediction of someone’s position. An accurate prediction is required to accomplish the purpose of the application that is to track one’s position from time to time.

Global Positioning System (GPS) is one of the systems that can be used to determine someone’s position. It can determine the position of devices equipped with a GPS signal receiver. The signals come from satellites that are orbiting the earth [2]. Nowadays, GPS is highly accurate [3]. However, GPS can only determine the position in an outdoor environment [2].

Indoor Positioning System (IPS) is used to overcome this problem. IPS works by utilizing available signals in the room. For example, Bluetooth and wireless LAN (WLAN). These devices generate signals. IPS uses the received signal strength index (RSSI) from signals that are generated by the devices [4]. Using RSSI, IPS can determine the unknown user’s position in an indoor environment [4].

The purpose of determining the position is to generate an accurate prediction. An accurate prediction is needed because the methods (GPS and IPS) are to predict the real position. Thus, a good method can determine the most accurate position.
In mobile applications, there is another factor that has to be considered besides the positioning accuracy. The factor is power. Power usage needs to be considered because mobile devices have a limited amount of power [5]. The most affecting power is computation complexity. Low complexity algorithm will use less power than high complexity algorithm.

There are two methods that IPS can use to determine position. They are pattern matching localization and path loss model.

Pattern matching is a fingerprinting method to predict user position in an indoor environment. This method predicts the position using signal strength patterns which are sampled through the training process (offline phase). The majority of received signal strength cannot be predicted because they fluctuate frequently [6]. So, this method can accurately predict the object’s position in an indoor environment by matching the nearest pattern of training data and testing data. The advantage of this method is it provides high indoor positioning accuracy [7]. However, the online phase suffers from high computation complexity \( O(n^p) \) [8].

The path loss model determines the unknown object position using the RSSI-distance equation [9]. Object distance can be determined by RSSI using the equation. The equation is generated through the training phase. Similar to pattern matching, this method uses RSSI values in its training phase. The model is generated using RSSI and its distance which are got from many samplings. This method utilizes trilateration to determine the unknown object’s position [10]. The advantage of this method is it has low computation complexity \( O(c) \) thus it is fast and only requires low power. So, it is suitable to be used on mobile devices. But, this method’s accuracy is still relatively low [11]. This happens because the indoor environment structure is complex [4].

In this paper, a new filter algorithm is proposed to improve path loss model accuracy. The proposed method works by replacing the RSSI value at a certain distance by considering the current and the previous RSSI value. The filtering process takes place in generating the model template or in the training phase.

The paper is arranged as follows. First, we show the related works that have been done. Second, we describe the proposed method. Third, we conduct and discuss the experiments. Lastly, we show the conclusion of this research.

2. Related Works
Some works have been done in improving Indoor positioning system accuracy. Researches based on pattern matching as follows. Oh and Kim [7] proposed an adaptive weighting algorithm to determine user position. The algorithm works by checking the Euclidean distance between reference points to generate the weight value. Zhong etc [12] proposed implementation of IPS based on k-Means Clustering. This algorithm’s accuracy is over 80%. Ma etc [13] proposed a weight fusion algorithm to improve the accuracy of IPS. The research considers the generated Euclidean distance, joint probability, and weight fusion algorithm to predict the user’s position. Song etc [14] proposed a weight distance fingerprint algorithm based on weighted K-NN to indoor-based applications. The algorithm calculates weight value based on RSSI variance and considers the distance between an access point and receiver device.

Researches based on the path loss model as follows. Sohan etc [2] proposed ITU indoor path loss model. The research predicts user position by considering positions from two setups that were installed in two different rooms. Wang and Luo [4] proposed a weighted centroid algorithm to remove the vertical distance effect (height) of the access point to the receiver device. Ma etc [15] proposed fingerprinting method bases on Bluetooth RSSI ranks. This research uses the Kendall Tau Correlation Coefficient to connect the newest signal position by signal strength ranks which are generated by iBeacon devices that are installed in a retail place.

3. Proposed Method
This paper is focused on improving path loss model accuracy in an indoor positioning system. The purpose of this research is to create a new filter algorithm in the generating model process in the path loss model.
There are two main steps in the proposed system which are shown in Figure 1. They are generating path loss models (training phase) and predicting position (testing phase). The training phase is conducted as follows. First, the training data is sampled using a mobile device (i.e. smartphone). Then, they are filtered using the new filter algorithm in the training phase. After that, the path loss models are constructed using the filtered data. The testing phase is conducted as follows. First, the testing data is sampled using a mobile device. After that, a trilateration algorithm calculates the unknown position using the processed testing data and the generated path loss models. Finally, the algorithm returns the predicted position to the user. It is given in coordinate \((x, y)\) form.

In summary, the training phase is generating models that are required in the positioning process. This phase returns path loss models. The testing phase is the predicting position process. This phase returns a prediction of the unknown position.

3.1. Training Phase: Generating Path Loss Model

The training phase is a required phase before the testing phase or predicting position process can be done. In this phase, path loss models are generated. They are generated using 3 steps: sampling training data, filtering training data, and generating models.

3.1.1. Sampling training data

Firstly, training data are sampled. This is the first step in the training phase. This step returns \(TD\) as the collection of the sampled training data. Training data are gathered in the sampling process. The process is conducted in the experiment area. The structure of train data is shown in (1).

\[
D_i = (SSID_i, RSSI_i, P_i)
\]  

(1)

One training data \(D_i\) represents a single sample of the entire training data. In a train data \(D_i\), there are \(SSID_i\), \(RSSI_i\), and \(P_i\). \(RSSI_i\) is the received signal strength index that is emitted by an access point that has a service set identifier \(SSID_i\). \(P_i\) is the device’s position while sampling data or sampling position. \(P_i\) is represented in coordinate format \((x, y)\) which represents the room’s coordinate in the Cartesian system.

The sampling process is done as follows. First, the user that is equipped with a mobile device moves to one of the sampling positions (e.g. \((1,1)\)) within the experiment area. In this position, the user samples the \(RSSI_i\) from \(SSID_i\) or train data \(D_i\) many times (e.g. 100 times). They are gathered using a mobile-based signal strength sampling application that is installed in the mobile device. Later, all of the gathered sample data \(D_i\) in this position will have the same \(SSID_i\) and \(P_i\). They will have different \(RSSI_i\)s but there is a high chance that the same \(RSSI_i\)s might also occur.

After the sampling is done, the sampling process is repeated in the same sampling position until training data \(D_i\) from all SSID in the room have been sampled. The sampling amount for each SSID is also the same as the sampling amount of the first SSID (e.g. 100 times).

After training data from all SSID in the room have been sampled, then the user moves to the next sampling position (e.g. \((1,2)\)). In this position, the sampling process is repeated all over again from the beginning until training data from all of the sampling positions have been sampled.

\[
TD = (D_i)_{i=1}^n, D_i \in TD
\]  

(2)
Lastly, all of the sampled training data $D_i$ are merged into a collection of training data $TD$. The structure of $TD$ is shown in (2). A $TD$ has $n$ amount of $D_i$ which is the total amount of the sampled data. $TD$ is the output of this step and is now ready to be used in the next step.

3.1.2. Preprocessing training data

Before the training data are filtered by the new filter algorithm, the training data $TD$ needs to be preprocessed. The purpose of this step is to get the distance $d_i$ between the sampling position $P_i$ and the access point $SSID_i$. This step will calculate $d_i$ and append it to the training data $D_i$. At the end of this step, every $D_i$'s structure in $TD$ will change into (3).

$$D_i = (SSID_i, RSSI_i, P_i, d_i)$$

The first train data $D_i$ is selected from $TD$. In $D_i$, there are $SSID_i$ and $P_i$. $RP_i$ is the position coordinate of the access point $SSID_i$. $RP_i$ is determined manually based on the experiment area. $d_i$ or $dp_{i,RP_i}$ is the distance between $P_i$ and $RP_i$. Here, $d_i$ is going to be calculated using the Euclidean distance formula (4).

$$d_{p_{i,RP_i}} = \sqrt{(x_{P_i} - x_{RP_i})^2 + (y_{P_i} - y_{RP_i})^2}$$

After $d_i$ has been calculated, it is appended into $D_i$. So, $D_i$'s structure now becomes (3). The next train data $D_i$ is selected from $TD$. The selected data $D_j$ has the same $P_i$ and $SSID_i$ as the previous data $D_i$. $D_j$ will have the same $d_j$ as $d_i$ because $P_j$ is same as $P_i$. So, $d_j$ is appended into $D_j$ and $D_j$'s structure now becomes (3). The next train data $D_k$ is selected from $TD$ and the process is repeated until all $D_k$ that has the same $P_i$ and $SSID_i$ has been exhausted.

Next, the next train data $D_i$ is selected from $TD$. Now, the selected $D_i$ is a train data whose $SSID_i$ is different from the first $SSID_i$. Because the $SSID$ is different, $d_i$ needs to be calculated. From here, the process is then repeated until all $SSID$ in the same $P_i$ has been exhausted. This will make sure that every $D_i$ will have the correct $d_i$ in respect to its $SSID_i$. After that, the next train data $D_m$ is selected from $TD$. The selected data $D_m$ has the next sampling position $P_m$ which different from $P_i$. Here, the process is repeated from the beginning until all $D_i$ in $TD$ has been processed.

Finally, all $D_i$'s structure has been changed into (3). Now, they have $d_i$ value. The preprocessed training data $TD$ is now ready to be used in the next step.

3.1.3. Filtering training data

Next, the collection of training data $TD$ is filtered before they are used as input for the model generation algorithm. This is the contribution of this paper. This step receives $TD$ as the input and returns a collection of filtered data $F_{TD}$ as the output.

$RSSI_{last}$ is an RSSI variable whose value is based on the $RSSI_i$ value from $D_i$. It is initially set with $RSSI_i$ value (5). It changes in every $D_i$ iteration (5) and is going to replace $RSSI_i$ in $D_i$.

$$RSSI_{last_i} = \begin{cases} RSSI_{last_{i-1}} + RSSI_{d_i}, & i = 1 \\ \frac{RSSI_{last_{i-1}} + RSSI_{d_i}}{2}, & i > 1 \end{cases}$$

The new filter algorithm works as follows. First, all $D_i$ in $TD$ is sorted based on its $d_i$. The data are sorted ascendingly from the lowest $d_i$ to the highest $d_i$. Next, the first $D_i$ is selected and $RSSI_{last}$ is set with $RSSI_i$ value. The next data $D_j$ that has the same $SSID_i$ is selected. Then, $RSSI_{last}$ is updated using the previous $RSSI_{last}$ and current $RSSI_i$ values (5). The updated $RSSI_{last}$ then replaces $RSSI_j$ value in $D_j$. $D_j$ becomes $F_{D_j}$ which is the processed $D_j$ whose $SSID$ has been replaced by $RSSI_{last}$. The next data $D_k$ that has the same $SSID_i$ is selected and the process is repeated until all $D_k$ that has the same $SSID_i$ is exhausted.
After that, the next data $D_i$ whose $SSID_i$ is different from $SSID_l$ is selected. $D_i$ has the lowest $d_i$, and is considered as the first training data. So, the filtering process is started again from the beginning. The process is repeated until all $D_i$ in $TD$ is exhausted.

$$F_{TD} = (F_{D_i})^n_{i=1}, F_{D_i} \in F_{TD}$$

(6)

The output of this step is a collection ($F_{TD}$) of the processed training data $F_{D_i}$. The structure of $F_{TD}$ is shown in (6). From (6), we can see that the total amount of $D_i$ in $F_{TD}$ is equal to the amount of the original data ($n$). This happens because the filtering algorithm does not delete any $D_i$ but replaces it. After $F_{TD}$ is generated, $F_{TD}$ is now ready to be used in the next step which is the template generation process.

3.1.4. Generating path loss models

Lastly, path loss models are created using $F_{TD}$. This is the last step in the training phase. This step receives $F_{TD}$ as the inputted training data. It returns the models in mathematical equation format.

3 models will be generated. The number 3 is picked because there must be a minimum of 3 models to make the trilateration (positioning) algorithm in the testing phase work. Each model is a mathematical equation. The general equation of the path loss model is shown in (7).

$$y = A \ln x + B$$

(7)

In (7), there are 2 known values. They are $x$ and $y$. The other variables $A$ and $B$ are constants and currently unknown. They are unknown and will be found by the model generation process. $x$ is the distance between the user or device and an access point. So, $x$ variable can also be called $d_{AP}$. $y$ is the received signal strength $RSSI_l$ that is emitted by the access point. So, $y$ variable can also be called $RSSI_{AP}$. For the sake of clarity, Eq. (7) can be changed into (8).

$$RSSI_{AP} = A \ln d_{AP} + B$$

(8)

The model generation process is run to find the unknown variables ($A$ and $B$) using the known variables ($RSSI_{AP}$ and $d_{AP}$) from the training data $F_{TD}$. The method works as follows. First, one of $F_{D_i}$ in $F_{TD}$ is processed. $F_{D_i}$'s structure is the same as $D_i$. So, a $F_{D_i}$ contains $SSID_i$, $RSSI_i$, $P_i$, and $d_i$ (3). From $F_{D_i}$, the method assigns $RSSI_i$ into $RSSI_{AP}$ and $d_i$ into $d_{AP}$. After the needed variables have been set, $RSSI_{AP}$ and $d_{AP}$ are assigned into (8). Next, the method iterates $F_{TD}$ and gets the next data $F_{D_j}$. Here, we need to note that $SSID_j$ and $P_j$ must be same as previous ($SSID_i$ and $P_i$) until instructed to change. This means that the method picks the next data $F_{D_j}$ with the same access point $AP$ and sampling position $P_j$ (e.g. (1,1)). But, $RSSI_j$ can be the same or different than $RSSI_i$. Using the same process, the method gets $RSSI_{AP}$ and $d_{AP}$ from $F_{D_j}$ and assigns them into (8). Using two equations of (8), the unknown constant values $A$ and $B$ can be determined.

The method then continues iterating $F_{TD}$ until all $F_{D_j}$ whose $SSID_j$ and $P_j$ are same as previous ($SSID_i$ and $P_i$) is exhausted. In every iteration, the method assigns them into (8) and evaluates the previous results to get more accurate $A$ and $B$ values. After that, the method iterates $F_{TD}$ with different sampling position $P_j$ (e.g. (1,2)) but still with the same $SSID_i$. The method repeats until all sampling position is exhausted. These repetitions ensure that every training data for a specific $AP$ are considered in the model.

Finally, the $AP$ path loss model is created by assigning $A$ and $B$ values into (8). The positioning algorithm can input the signal strength $RSSI$ into the constructed model and it will return the distance.

Until now, there is only one generated model. Two more models need to be generated because the predicting process (trilateration) uses at least 3 models to be able to determine an unknown position. To make 2 more path loss models, here the method repeats once again from the beginning using the next different $SSID_j$ and a new (8) model to remove all previous $SSID_i$ calculation because 1 $AP$ can only have 1 path loss model.

$$C_{model} = (RSSI_{AP,1}, RSSI_{AP,2}, RSSI_{AP,3})$$

(9)
After the models have been created, they are packed into a container $C_{\text{model}}$ as shown in (9). $C_{\text{model}}$ contains 3 path loss models and is now ready to be used in the testing phase or predicting position.

3.2. Testing Phase: Predicting Position

The testing phase is the position prediction phase after the training phase has been done. In this phase, the unknown user’s position is predicted using the generated path loss models. There are 3 steps in this phase: sampling testing data and determining the position.

3.2.1. Sampling testing data

The first step in determining position is to get the testing data. Testing data are got by sampling the test data. The structure of the test data is shown in (10). The output of this step is a collection of the sampled test data $TD'$.

$$D'_i = (SSID'_i, RSSI'_i)$$  \hspace{1cm} (10)

A test data $D'_i$ contains two data: $SSID'_i$ and $RSSI'_i$. $SSID'_i$ is the service set identifier of the access point that emits signal strength index $RSSI'_i$. The difference between test data $D'_i$ and the train data $D_i$ is $D'_i$ does not have $P_i$ data. In the testing phase, $P_i$ is the position of the current user position $TP$. $TP$ is unknown and is the main variable that is going to be calculated in this phase.

The sampling process is done as follows. In $TP$, the user who is equipped with the sampling device samples the test data $D'_i$. The user samples $RSSI'_i$ from one of the access point $SSID'_i$. The sample $D'_i$ is taken many times (e.g. 10 times) using a mobile-based sampling application. The gathered sample data $D_i$ will have the same $SSID'_i$ and $RSSI'_i$. But, different $RSSI'_i$ might also occur.

After that, the sampling process is repeated but using the next SSID $SSID'_i$ until the $RSSI'_i$sample from all of the $SSID'_i$ in the room have been gathered. The user also records the $TP$ while it is not included in the test data. $TP$ will be used in evaluating the experiment result.

$$TD' = (D'_i)_{i=1}^n, D'_i \in TD'$$  \hspace{1cm} (11)

Finally, same as the training step, all of the sampled test data $D'_i$ are combined into a collection of training data $TD'$ whose structure is shown in (11). $TD'$ has $n$ amount of test data $D'_i$. $TD'$ is now ready to be used.

3.2.2. Determining the position

The next step is to determine the position based on the generated models $C_{\text{model}}$ and the test data $TD'$. This step receives $C_{\text{model}}$ and $TD'$ as the input and returns the prediction of the unknown user position $TP$ as the output.

First, the first test data $D'_1$ is selected from $TD'$. From $D'_1$, there are access point whose service set identifier is $SSID'_1$ and test signal strength $RSSI'_1$. Then, a path loss model $RSSI_{SSID'_1}$ from $C_{\text{model}}$ is selected. $RSSI_{SSID'_1}$ is chosen based on $SSID'_1$ because $RSSI_{SSID'_1}$ is the path loss model of $SSID'_1$ access point. Here, the distance $d_{AP}$ between $TP$ and the reference point $RP_{AP}$ of the access point $AP$ is going to be calculated. After that, $RSSI'_1$ is entered into the model’s $RSSI_{SSID'_1}$ formula (8). Later, (8) will return $d_{SSID'_1}, d_{SSID'_1}$, and $d_{SSID'_1}$ which is the normalized distance between $TP$ and $RP_{SSID'_1}$.

The process is repeated from the beginning after $d_{SSID'_1}$ is calculated. The next test data $D'_2$ is selected from $TD'$. Now, $D'_2$ whose $SSID'_2$ is different from the previous SSID $SSID'_1$ is selected. The different SSID is chosen because the next distance $d_{SSID'_2}$, needs to be calculated. $d_{SSID'_2}$ is the distance between $TP$ and the position of the next SSID $SSID'_2$ $RP_{SSID'_2}$. The same process is repeated once again so now there are 3 distances $d_{SSID'_1}, d_{SSID'_2},$ and $d_{SSID'_3}$.

Finally, the distances $d_{SSID'_1}, d_{SSID'_2},$ and $d_{SSID'_3}$ are packed into one collection $C_d$ (12). Here, $C_d$ is now available but $TP$ is still not available. The next method is required to determine the exact value of $TP$.

$$C_d = (d_{AP_1}, d_{AP_2}, d_{AP_3})$$  \hspace{1cm} (12)
Trilateration is a method that is used to determine the unknown position using the known reference points’ positions. The unknown position can be determined if there are minimal 3 known reference points and 3 known distances between the unknown position to every reference point. Trilateration converts the distances $d_i$ inside $C_d$ into the unknown user position $TP$ which is in a coordinate form $(x, y)$. It takes $C_d$ as the input and returns the predicted position $TP$ as the output.

The illustration of how trilateration works is visualized in Figure 2. Red dots are the reference points. A reference point $RP_{AP}$ is the position where the access point $AP$ is located and is represented in coordinate form $(x, y)$. There must be minimal of 3 reference points known to make the trilateration method works. Every $RP_{AP}$ should have been known from the very beginning. They can be got by manually examining the position of the $AP$ based on the map of the experiment area. A reference points container $C_{RP}$ (13) is used to contain these 3 reference points.

$$C_{RP} = (RP_{AP_1}, RP_{AP_2}, RP_{AP_3})$$ (13)

Yellow dot $TP$ is the user or device position. It still currently unknown and will be predicted in this phase. $d_{AP}$ is the distance between $TP$ and reference point $RP_{AP}$. Every $d_{AP}$ is related to $RP_{AP}$ and has been calculated in the previous step. So, now there is $C_d$ which contains 3 $d_{AP}$s.

Figure 2 also illustrates the possible user’s position influenced by his/her $d_{AP}$. For example, examine $d_1$. The big green circle created by $d_1$ is the possible position. The user might be anywhere in the circle. Here, the method’s purpose is to determine the user’s exact position $TP$. So, the second reference point $RP_2$ is introduced. $d_2$ also creates the big pink circle where represents all of the user’s possible position. But, using $RP_1$, $d_1$, $RP_2$, and $d_2$, user’s possible position can be reduced into 2 positions. They are located at the intersection of the big green circle and the big pink circle. To find $TP$, the third reference point $RP_3$ is introduced. Using the same way and Euclidean distance formula (4), the exact $TP$ position can be determined. This is the reason there must be 3 $RP_{AP}$ and 3 $d_{AP}$ minimal in order to make the trilateration method be able to calculate $TP$.

The variables that are currently available to be used are $C_{RP}$ and $C_d$. $C_{RP}$ contains $RP_{AP_1}$, $RP_{AP_2}$, and $RP_{AP_3}$ which fit into $RP_1$, $RP_2$, and $RP_3$ in Figure 2. $C_d$ contains $d_{AP_1}$, $d_{AP_2}$, and $d_{AP_3}$ which fit into $d_1$, $d_2$, and $d_3$ in Figure 2. Using these variables and the trilateration method that has been described, $TP$ can now be obtained. $TP$ is the result of the predicted user’s position. It is the very final output of the predicting position phase of the testing phase. The phase ends here because $TP$ has been obtained.

Figure 2. Trilateration Illustration
4. Experiment and Analysis
An experiment is conducted to check the effectiveness of the proposed method. The proposed method will be compared with the normal path loss model method. The parameters that will be measured are accuracy and performance. The accuracy parameter is used to determine how much accuracy is improved in the proposed method. Performance parameter is used to determine how robust the proposed method is compared to the existing method. The experiment was conducted in Windows 10 64-bit system powered by a 2.5 GHz AMD A12-9700P CPU. The used programming language is Python version 3.9.1.

4.1. Experiment Area
Before the training phase begins, the experiment area is determined. The area will be used to sample the training and testing data. It will also be used as a reference by the proposed method for its prediction output. The criteria of the experiment area are indoor environment, has human activity, and has 3 wireless LANs (WLANs). So, a family room of the author’s house is used. Its dimension is 7 × 7 meters.

![Figure 3. Experiment Area](image)

The floor plan of the experiment area is shown in Figure 3. Red dots are the sampling points. The points are defined as area coordinate in x and y value. The top-left red dot is the first point which is in (1,1). The bottom-right red dot is the last point which is in (6,6). These points are used to sample the training and testing data. They are also used as the coordinate of the predicted object position.

Yellow dots are the position of the WLAN devices. They are used as reference points. The reference points are also used as the sampling points too. So, there will be zero meter distance between the sampling point and the access point (reference point) in the training and testing data.

In each of the sampling points, both training data and testing data are sampled 200 times. Based on Figure 3, there are 3 access points and 33 sampling points. So, the total amount of the sampled data are 19,800. The data are sampled using an Android-based application in Xiaomi Mi4i devices. There is a 3 seconds delay in each of the samples. As in (1), each of the sampled data contains SSID, RSSI, and sampling point coordinate. Then, the sampled data are shuffled and divided into two: 100 training data and 100 testing data. Training data are supplied into the training phase. Testing data are supplied into the testing phase. The mean of 10 random data from the testing data is used as the input of test data. In the testing phase, the sampling point data will be used as the ground truth for the accuracy comparison.

4.2. Generated Path Loss Models
Based on the training data, path loss models are generated in the training process. They are mathematical equations based on (8). Each access point has one model. Based on the experiment area, there are 3 generated models. Equations (14), (15), and (16) show the models for each access point.
Figure 4. Models Generated By The Proposed Method

\[ RSSI_{\text{WiredBB8}} = -10.7385 \ln(d_{\text{WiredBB8}}) - 39.3742 \]  
\[ RSSI_{\text{Realme AP 1}} = -20.3638 \ln(d_{\text{Realme AP 1}}) - 38.4406 \]  
\[ RSSI_{\text{Realme AP 2}} = -16.333 \ln(d_{\text{Realme AP 2}}) - 38.2789 \]  

The graph visualization of the models is shown in Figure 4. In Figure 4, the models' mathematical equations are represented as lines. The dots are the filtered training data that are being used as reference points to build the models.

4.3. Accuracy

Accuracy comparison is provided to check the accuracy of the proposed method. The accuracy is calculated by calculating the error mean (in meter) of the predicted position concerning its ground truth. The result of the accuracy comparison is shown in Table 1. Green cell means the predicted position’s error of one’s method is lower than the other method which is shown in the orange cell. The proposed method gives 6 accuracy improvements, 1 accuracy reduction, and 24 unchanged accuracy or position predictions compared to the existing method.

| Ground Truth | Existing Method | Proposed Method |
|--------------|----------------|-----------------|
|              | Predicted Position | Error (meter) | Predicted Position | Error (meter) |
| (1,1)        | (2,6)            | 5.1            | (2,5)            | 4.1           |
| (2,1)        | (3,4)            | 3.2            | (4,3)            | 2.8           |
| (3,1)        | (4,3)            | 2.2            | (4,3)            | 2.2           |
| (4,1)        | (4,3)            | 2              | (4,3)            | 2             |
| (1,2)        | (4,3)            | 3.2            | (4,3)            | 3.2           |
| (2,2)        | (4,3)            | 2.2            | (4,3)            | 2.2           |
| (3,2)        | (4,3)            | 1.4            | (4,3)            | 1.4           |
| (5,2)        | (4,3)            | 1.4            | (4,3)            | 1.4           |
| (1,3)        | (2,4)            | 1.4            | (2,4)            | 1.4           |
Table 1 shows that the average error of the proposed method (1.89 meters) is lower than the existing method (2.1 meters). So, the average accuracy of the proposed method is 3% higher than the existing method. The proposed method has higher accuracy because the combination of distances that are returned by the proposed models is different from the existing method one. Based on the combination, trilateration gives a more accurate predicted position. Thus, the predicted position that is returned by the proposed method has better accuracy.

4.4. Performance

In addition to this research, a performance comparison is also provided to check the robustness of the proposed method. In the training phase, the performance is calculated by getting the computation time of the model generation step. The train data are already available. In the testing phase, the performance is calculated by getting the computation time of the predicting process to get the user’s position. The models and test data are already available. The data that are available are not being calculated again. The performance result is shown in Table 2.

| Phase | Mean Computation Time (ms) |
|-------|---------------------------|
|       | Existing Method | Proposed Method |
| Training | 1               | 1               |
| Testing  | 267             | 255             |

Table 2. Performance Comparison
From Table 2, we can see that the computation time of the proposed method and the existing method. In the training phase, the proposed method (1 ms) is similar to the existing method (1 ms). The reason is the amount of the training data is relatively low (100) from the processor’s computation speed perspective. In the testing phase, the proposed method computation time (255 ms) is relatively similar to the existing method (267 ms). The reason is they are using the same positioning algorithm.

5. Conclusion
Research to improve indoor positioning system accuracy has been conducted. To improve accuracy, a new filter algorithm is introduced. It filters the received signal strength index (RSSI) in the training phase. By considering the previous and current value, RSSI at a certain distance is replaced by a new one. The experimental result shows that the proposed method has 3% better accuracy than the existing method.

References
[1] Zhu D, Zhao B and Wang S 2018 Mobile target indoor tracking based on Multi-Direction Weight Position Kalman Filter Comput. Networks 141 115–27
[2] Sohan A A, Ali M, Fairooz F, Rahman A I, Chakrabarty A and Kabir M R 2019 Indoor positioning techniques using RSSI from wireless devices 2019 22nd Int. Conf. Comput. Inf. Technol. ICCIT 2019 18–20
[3] Naik R K and Nair B B 2019 Improving GPS based distance measurement accuracy using Machine Learning: An empirical study Proc. 4th Int. Conf. Commun. Electron. Syst. ICCES 2019 1294–9
[4] Wang P and Luo Y 2017 Research on WiFi indoor location algorithm based on RSSI Ranging Proc. - 2017 4th Int. Conf. Inf. Sci. Control Eng. ICISCE 2017 1694–8
[5] Bisio I, Lavagetto F, Marchese M and Sciarrone A 2016 Smart probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices Pervasive Mob. Comput. 31 107–23
[6] Lukito Y and Chrismanto A R 2017 Recurrent neural networks model for WiFi-based indoor positioning system Proceeding 2017 Int. Conf. Smart Cities, Autom. Intell. Comput. Syst. ICON-SONICS 2017 2018-Janua 121–5
[7] Oh J and Kim J 2018 Adaptive K-nearest neighbour algorithm for WiFi fingerprint positioning ICT Express 4 91–4
[8] Du Y and Arslan T 2018 A segmentation-based matching algorithm for magnetic field indoor positioning 2017 Int. Conf. Localization GNSS, ICL-GNSS 2017 1–5
[9] Adame T, Carrascosa M and Bellalta B 2019 The TMB path loss model for 5 GHz indoor WiFi scenarios: On the empirical relationship between RSSI, MCS, and spatial streams IFIP Wirel. Days 2019-April 1–8
[10] Grzechca D E, Pelezar P and Chruszczyn L 2016 Analysis of Object Location Accuracy for iBeacon Technology based on the RSSI Path Loss Model and Fingerprint Map Int. J. Electron. Telecommun. 62 371–8
[11] Zhao B, Zhu D, Xi T, Jia C, Jiang S and Wang S 2019 Convolutional neural network and dual-factor enhanced variational Bayes adaptive Kalman filter based indoor localization with Wi-Fi Comput. Networks 162
[12] Zhong Y, Wu F, Zhang J and Dong B 2017 WiFi indoor localization based on K-means ICALIP 2016 - 2016 Int. Conf. Audio, Lang. Image Process. - Proc. 663–7
[13] Ma R, Guo Q, Hu C and Xue J 2015 An improved WiFi indoor positioning algorithm by weighted fusion Sensors (Switzerland) 15 21824–43
[14] Song X, Yang F, Ding L and Qian L 2012 Weight adjust algorithm in indoor fingerprint localization 6th Int. Conf. Signal Process. Commun. Syst. ICSPCS 2012 - Proc.
[15] Ma Z, Poslad S, Bigham J, Zhang X and Men L 2017 A BLE RSSI Ranking based Indoor Positioning System for Generic Smartphones