IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 31 July 2019, accepted 22 August 2019, date of publication 5 September 2019, date of current version 14 November 2022.

Digital Object Identifier 10.1109/ACCESS.2019.2939545

How to Get the Best Out of Your Fingerprint Database: Hierarchical Fingerprint Indoor Positioning for Databases With Variable Density

QIANG CHANG[1], (Senior Member, IEEE), SAMUEL VAN DE VELDE[2,3], (Senior Member, IEEE), AND HEIDI STEENDAM[2], (Senior Member, IEEE)

[1]National Innovation Institute of Defense Technology, Beijing 100000, China
[2]Department of Telecommunications and Information Processing, Ghent University, 9000 Ghent, Belgium
[3]Pozyx, 9000 Ghent, Belgium

Corresponding author: Heidi Steendam (heidi.steendam@ugent.be)

This work was supported in part by the Belgian National Fund for Scientific Research (FWO Flanders), and in part by the Natural Science Foundation of China (NSFC) Program under Grant 51675525. The work of Q. Chang was supported by the China Scholarship Council (CSC).

ABSTRACT In this paper, we consider wireless positioning using Received Signal Strength (RSS) fingerprinting. To obtain good accuracy, this technique requires a database containing a high density of up-to-date fingerprints. However, as acquiring fingerprints through training is labor intensive and the indoor topology is subject to changes, a high density fingerprint database cannot always be obtained. On the other hand, the time to retrieve data from a database with high density can be too high for real-time positioning. To tackle these issues, we introduce the Hierarchical Positioning Algorithm (HPA). In this algorithm, we divide the database into a number of sub-databases with different densities, each containing a sufficiently small number of fingerprints to reduce the data retrieval time. The algorithm starts with a coarse estimate at the highest level, and gradually improves the accuracy in going to the lowest level. This HPA technique requires the construction of sub-databases containing fingerprints that are properly selected to obtain the wanted level of accuracy. This paper considers two algorithms to construct the database: the Minimum Distance Algorithm (MDA) to select the reference points, and the Local Gaussian Process (LGP) algorithm to determine the RSS values at the selected reference points. Simulation results show that the hierarchical algorithm, combined with MDA and LGP to construct the sub-databases, is a fast algorithm that can achieve high accuracy, even with a database having a variable density of fingerprints.

INDEX TERMS Indoor positioning, signal fingerprint, Gaussian process, discrete level of detail.

I. INTRODUCTION

The existence of an accurate indoor positioning system is a prerequisite for Location Based Services (LBS), which are implemented in an increasingly number of applications. However no accurate large scale positioning system is available yet, due to the lack of infrastructure, the high hardware cost, or the low accuracy of the solutions. One of the most promising low-cost solutions is based on RSS fingerprinting. In this approach, the position is estimated by comparing the Received Signal Strength (RSS) with fingerprints in a database. A fingerprint indoor positioning system consists of two phases. In the first, off-line training phase, a database of position-fingerprints is constructed by inserting for a number of reference points (RPs) with the received signal strength (RSS) from different signal sources, such as WiFi access points [1], FM [2], UWB(ultra wide band) [3] and geomagnetic fields [4]. In the second, on-line positioning phase, the RSS values from different sources, measured by the user, are compared with the data in the database, and based on the best match, the user's position is determined. Because of the widespread availability of different signal sources, the deployment cost is low. Further, the algorithms can easily be implemented in mobile devices, and result in

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

a reasonable accuracy. Currently, some commercial fingerprinting implementations are available, such as Google Map Indoor [5], WiFiSlam [6] or Rtmap [7].

The main issue in fingerprint positioning is the requirement of a high-density up-to-date database, as the accuracy of the fingerprinting technique highly depends on the density and accuracy of the fingerprints in the database. Firstly, as constructing and maintaining a high-density database is labor intensive, expensive and sometimes practically impossible, because of the complex local environments, the available databases show diverse training data densities, and sometimes insufficient data for accurate positioning. As point-by-point measurements of the fingerprints is too labor intensive and expensive, alternative solutions to construct and maintain a high-density database were proposed in the literature. The first solution is crowdsourcing [8]. In crowdsourcing, the user upload their fingerprints to update the database. However, encouraging the user to upload their data is a challenge. The second solution is creating a virtual database using mathematical model, such as linear and exponential model [9], MotleyKeenan model [10], Log-Distance Path Loss (LDPL) model [11], Gaussian processes (GP) model [12]–[14] and so on. The third way is ray-tracing [15]. In ray-tracing, a detailed description of the environment is required to build the fingerprint database. The fourth method is simultaneous localization and mapping (SLAM) [16]. In SLAM, the users are equipped with a receiver and an IMU, and the database is populated on the fly. As the developement of deep learning, the researchers are trying to present the use of deep neural networks (DNN) for WiFi fingerprinting [17]–[20]. The fingerprint database is replaced by a deep neural network. However, training the deep neural network requires a lot of labeled data.

The second issue in fingerprint positioning is the time required to search the database, which is related to the centralized nature of the fingerprint position algorithms. Because of the memory required for a dense database, the database is generally stored in a central location server. Every time a user wants to estimate its position, the user first needs to collect accurate measurements of the RSS and submit the measured values to the location server. The server searches the database to find the most likely match, from which the user's position is derived. As data retrieval from a large database is time-consuming, the users might not receive their results in real time.

To cope with the issues in fingerprint positioning, we introduce the Hierarchical Positioning Algorithm (HPA), which is based on the Discrete Level of Detail (DLOD) algorithm [21] from computer graphics. This HPA algorithm, however, requires the construction of a hierarchical database for the different levels of the algorithm, containing different densities of fingerprints. In order to have the same accuracy over the whole area, the fingerprints in each sub-database of the hierarchical database must have a spatial distribution that is as uniform as possible. For constructing the hierarchical database from the variable density database, we use the Minimum Distance Algorithm (MDA) in order to select the best RPs and the Local Gaussian Process (LGP) algorithm to estimate the RSS values at the selected RPs. Compared with GP, LGP has lower complexity, at the cost of a slight deterioration of the accuracy. The HPA starts with a coarse estimate at the highest level, and gradually improves the accuracy in going to the lowest level. In this paper, we show that the proposed HPA algorithm, with the hierarchical database constructed using LGP and MDA, improves the accuracy in areas with sparse training data, and reduces the time-consumption of the position estimation in dense data areas. To the best knowledge of the authors, this article is the first presenting the use of DLOD for fingerprint positioning.

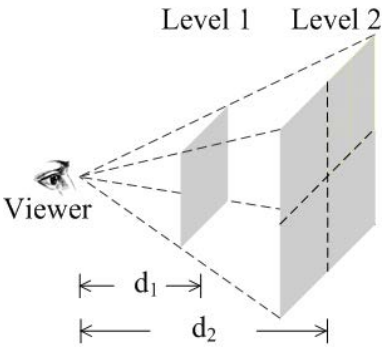


FIGURE 1. The basic idea of DLOD.

The rest of the paper is organized as follows. Section II introduces the hierarchical positioning algorithm, and reviews the MDA and LGP algorithms. Section III provides the simulation results and the conclusions are given in Section IV.

II. THE HIERARCHICAL POSITIONING ALGORITHM

A. SYSTEM DESCRIPTION

In this paper, we concentrate on 2D positioning. Assume the fingerprint database covers an area P of $S(m^2)$, and contains fingerprints from a signal sources, such as FM, WiFi, DVB and DTMB. During the training phase, we measure signal strengths at n reference points (RPs) $RP_i = (x_i, y_i), i = 1, 2, \cdots, n$, and store the resulting RSS values together with the coordinates. Based on the resulting training database DB_{tr}, we will estimate a user's position during the on-line phase.

In order to reduce the time for searching the database, we propose the Hierarchical Positioning Algorithm (HPA). This algorithm is based on the Discrete Level of Detail (DLOD) algorithm used in computer vision to decrease the complexity of representation of a 3D object when it moves away from the viewer [22]. Figure 1 illustrates the basic idea of DLOD.

In Fig. 1, if we are far from the object, we just need to render level 1, which contains fewer details. When we move closer, then we render one of the four tiles in Level 2 that contains more details according to our view. Using DLOD is a way of decrease the complexity of a 3D object representation.

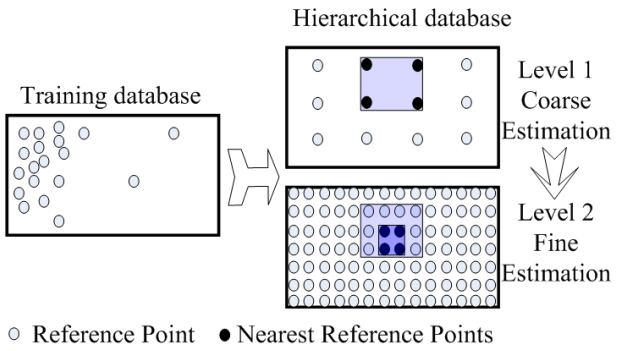


FIGURE 2. A hierarchical database with two levels.

In the HPA algorithm, we use a hierarchical database containing different levels. In a high-level sub-database, the area P is sparsely covered with a low density of fingerprints, resulting in a coarse estimate. At a low-level sub-database, the data is dense and results in a fine estimate. Figure 2 illustrates a hierarchical database with two levels. Let us denote the database at level k as $DB^{(k)}$. We assume database $DB^{(k)}$ contains $n^{(k)}$ reference points $RP_i^{(k)}$, $i = 1, \ldots, n^{(k)}$ with coordinates $(x_i^{(k)}, y_i^{(k)})$ and RSS values $\{(RSS_{i,j}^{(k)}, \delta_{i,j}^{(k)})|j = 1, 2, \cdots, a_i\}$, where a_i is the number of signal sources within $RP_i^{(k)}$'s visibility, $RSS_{i,j}^{(k)}$ is the measured signal strength corresponding to signal source j and $\delta_{i,j}^{(k)}$ is the uncertainty on the RSS value. In order to obtain the wanted accuracy over the whole area P, the RPs at each level have to be spatially uniformly distributed. However, the database constructed in the training phase will in practice have variable density, such that extracting the hierarchical database from the training database is not straightforward. In order to build the hierarchical database, we consider the minimum distance algorithm (MDA) to select the RPs as uniformly as possible over the area, and the local Gaussian process (LGP) algorithm to estimate the RSS values for the selected RPs. After the hierarchical database is constructed, we propose an altered version of the K-weighted nearest neighbors (KWNN) algorithm to extract the user's position from the hierarchical database.

B. MINIMUM DISTANCE ALGORITHM

The hierarchical fingerprint database consists of several sub-databases with different densities of fingerprints. As stated earlier, the fingerprints in each sub-database should be selected as uniformly as possible over the area P. Define $n^{(k)}$ as the number of reference points to be selected for the database $DB^{(k)}$ at level k. However, for general values of $n^{(k)}$, it is not straightforward to uniformly distribute the RPs over the area. Therefore, we propose a low-complexity algorithm to select the positions of the RPs: the Minimum Distance Algorithm (MDA). In this algorithm, the selection of the positions of the RPs of the sub-database at level k is based on a virtual sample database $DB_v^{(k)}$, which is constructed by placing a square grid in the area P with grid size $\lambda^{(k)}$, where the positions of the virtual RPs are selected as the corners of the squares in the grid. Assuming the room has size $x_{\max} \times y_{\max}$, the number of virtual positions equals $\lfloor \frac{x_{\max}}{\lambda^{(k)}} \rfloor \cdot \lfloor \frac{y_{\max}}{\lambda^{(k)}} \rfloor$. The $n^{(k)}$ positions of the RPs for sub-database k are selected out of the virtual sample database $DB_v^{(k)}$. We initialize the algorithm by randomly choosing one virtual position RP_c from $DB_v^{(k)}$: $DB^{(k)} = \{RP_c\}$. The other $n^{(k)} - 1$ positions are picked from the virtual database $DB_v^{(k)}$ based on the measure function

$$M_i = \sum_j \frac{1}{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1}$$

where (x_i, y_i) is the coordinate of the candidate position $\in DB_v^{(k)}$ and (x_j, y_j) are the coordinates of the RP positions already present in the sub-database $DB^{(k)}$. The virtual position that minimizes M_i is selected and added to the database $DB^{(k)}$. Because the measure function M_i is inversely proportional to the Euclidean distances between the candidate RF and the RPs in the database $DB^{(k)}$, candidate positions that are far from the already selected RP positions are favored, while candidate positions near already selected RP positions are filtered out. As a result, the distances between the RPs will be maximized and the RPs in $DB^{(k)}$ will be distributed uniformly and expand to the very edges of the area P. The algorithm is shown in Algorithm 1.

Algorithm 1 *MDA*

Require: the area P, the distance $\lambda^{(k)}$ between neighbor virtual RPs in $DB_v^{(k)}$, the number $n^{(k)}$ of RPs we want to select.
Ensure: select RPs every $\lambda^{(k)}$ meters in P to build $DB_v^{(k)}$,
Ensure: randomly select RP_c from $DB_v^{(k)}$, $DB^{(k)} = \{RP_c\}$
  while $|DB^{(k)}| \neq n^{(k)}$ **do**
    for all $RP_i \subset DB_v^{(k)}$ **do**
      Calculate M_i (1)
    end for
    $RP = \arg \min_{RP_i \in DB_v^{(k)}} M_i$
    $DB^{(k)} \leftarrow RP$
  end while

To illustrate MDA, we consider an area P of $16.5m \times 48.5m$, and $\lambda^{(k)} = 0.5m$. Figure 3 shows the positions of the RPs in $DB^{(k)}$ when $n^{(k)} = 20$ and 40 RPs are selected out of the virtual database $DB_v^{(k)}$. Further, the figure shows the positions of the RPs when the RPs are selected randomly from $DB_v^{(k)}$. As can be observed, the proposed algorithm is able to select the RPs spatially uniform over the area P.

After the positions of the RPs in database $DB^{(k)}$ are selected with MDA, the RSS values for these RPs need to be determined. To this end, we compare the positions of the RPs in $DB^{(k)}$ with those in the training database DB_{tr}. Whenever one or more RPs in DB_{tr} are within a distance $\varepsilon^{(k)}$ of a RP RP_i in $DB^{(k)}$, we will replace the position of the RP in $DB^{(k)}$ with the position of the nearest RP in DB_{tr}, together with its RSS values and the uncertainty on the measured

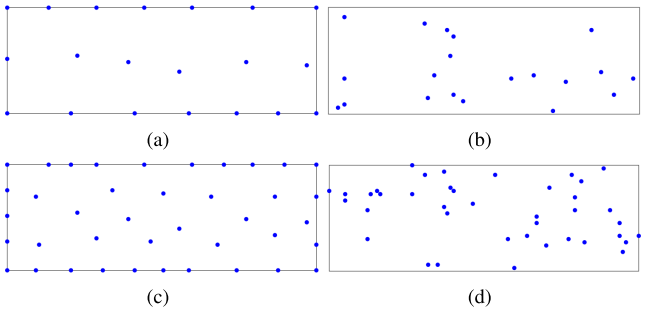


FIGURE 3. Positions of the RPs (a) MDA, $n^{(k)} = 20$ (b) MDA, $n^{(k)} = 40$ (c) randomly, $n^{(k)} = 20$ (d) randomly, $n^{(k)} = 40$.

RSS values. If no RPs in DB_{tr} are within a distance $\varepsilon^{(k)}$ of a RP RP_i in $DB^{(k)}$, the Local Gaussian Process (LGP) algorithm will be used to estimate the RSS values and their uncertainty in RP_i.

The resulting sub-database $DB^{(k)}$ is determined by three parameters: the number $n^{(k)}$ of RPs, the distance $\lambda^{(k)}$ between RPs in the virtual sample database and the radius $\varepsilon^{(k)}$ within which is looked for nearby training RPs. The number $n^{(k)}$ of RPs is defined by the accuracy that is targeted at level k of the hierarchical positioning algorithm. The distance $\lambda^{(k)}$ determines not only the spatial uniformity of the resulting RPs, but also the complexity of the algorithm: by reducing $\lambda^{(k)}$, the RPs will be placed more uniformly over the area P, but the complexity of MDA increases as the number of virtual RPs to be searched increases in an inverse proportion to the quadrate of $\lambda^{(k)}$. Finally, the radius $\varepsilon^{(k)}$ will have an influence on the performance of the HPA. When the radius is small, the resulting database $DB^{(k)}$ will have a more uniform placement of the RPs, but the probability of finding a nearby training RP decreases, such that the RSS of more RPs need to be determined using the LGP algorithm. On the other hand, when the radius is selected large, the resulting database $DB^{(k)}$ will be less spatially uniform, but more training RPs will be present in $DB^{(k)}$. In the numerical results, we will evaluate the effect of the parameters $\lambda^{(k)}$ and $\varepsilon^{(k)}$ on the performance of the positioning.

C. THE LOCAL GAUSSIAN PROCESS ALGORITHM

The Local Gaussian Process (LGP) algorithm is used to reduce the computational complexity of the Gaussian Process (GP) algorithm, which is used to predict unknown RSS values at positions that are not in the training database. In [12], various GP sparsification methods are introduced. LGP is proposed from physical point of view, while various GP sparsification methods in [12] are introduced from a mathematical point of view.

In this section, we first review the GP algorithm. This algorithm starts from the property that RSS values at surrounding positions are correlated. Because of this correlation, it is possible to describe the RSS at positions where the RSS is not known as function of the RSS at positions where the RSS value is measured. The GP algorithm uses the Gaussian kernel to describe this correlation. As a result, the correlation matrix between the noisy RSS values RSS_i at positions $\mathbf{c}_i = \{x_i, y_i\}$, $i = 1, \ldots, n$, measured during the training phase, can be written as:

$$cov\rho = \mathbf{Q} + \mathbf{S} \tag{2}$$

where $\rho(i) = RSS_i$, $\mathbf{Q}_{i,j} = k(\mathbf{c}_i, \mathbf{c}_j)$, and $\mathbf{S} = diag\{\sigma_i^2\}$ is the diagonal matrix of the variances of the measured RSS values RSS_i [13]. Further, $k(\mathbf{c}_i, \mathbf{c}_j)$ is the Gaussian kernel function:

$$k(\mathbf{c}_i, \mathbf{c}_j) = \sigma_f^2 \exp(-\frac{1}{2l^2}||\mathbf{c}_i - \mathbf{c}_j||^2) \tag{3}$$

where σ_f^2 and l are the signal variance and length scale, respectively, determining the correlation with the RSS values at surrounding positions. The parameters σ_f^2, and l can be estimated using hyper-parameter estimation [13]. This covariance matrix can be used to predict the RSS value RSS_* at an arbitrary position $\mathbf{c}_* = \{x_*, y_*\}$. The posterior distribution of the RSS value at any position is modeled as a Gaussian random variable, i.e. $(RSS_*|\mathbf{c}_*) = \mathcal{N}(RSS_*; \mu_*, \sigma_*^2)$, where μ_* and σ_*^2 are given by:

$$\mu_* = \mathbf{k}_*^T(\mathbf{Q} + \mathbf{S})^{-1}\rho \tag{4}$$

$$\sigma_*^2 = k(\mathbf{c}_*, \mathbf{c}_*) - \mathbf{k}_*^T(\mathbf{Q} + \mathbf{S})^{-1}\mathbf{k}_* + \sigma_n^2 \tag{5}$$

where σ_n^2 is the measurement variance, and $\mathbf{k}_*(i) = k(\mathbf{c}_*, \mathbf{c}_i)$, $i = 1, \ldots, n$. The estimate of the RSS value at position $\mathbf{c}_* = \{x_*, y_*\}$ equals $RSS_* = \mu_*$ and the uncertainty on the estimated RSS is σ_*^2.

For a large area containing several hundreds of RPs, computing the RSS values with equation (4) and (5) are computationally demanding because of the inversion of the large covariance matrix (2). However, in an indoor environment, we may assume that RPs at a large distance from the position where we want to estimate the RSS value, are blocked by several walls and other objects. Hence, the covariance $k(\cdot, \cdot)$ between the RSS value of those far away RPs and the RSS value at the considered position will be approximately zero. As a result, it is a reasonable assumption that only training RPs close to the considered position will contribute to the RSS value at considered position. The LGP algorithm restricts the training RPs that contribute to the RSS value at position $\mathbf{c}_*$ to a training set TS_*, setting $k(x_*, x_i) = 0$ if $x_i \notin TS_*$. Assuming the number of RPs in TS_* equals L, the LGP algorithm simplifies equation (4) and (5) by only considering the k nearest RPs. i.e. $\mathbf{k}_*$ and ρ reduce to a $L \times 1$ vector, and $cov\rho$ (2) to a $L \times L$ matrix. Compared to the complexity $\mathcal{O}(n^3)$ when all n RPs in the training database are used, the LGP algorithm has complexity $\mathcal{O}(nL)$ to select the L nearest RPs and $\mathcal{O}(L^3)$ to invert the reduced-size covariance matrix (2).

To illustrate the LGP algorithm, we consider the RSS radio map of a WiFi access point in an indoor environment. The true radio map is created using the WinProp tool from AWE Communications [23]. The area is a $19.5m \times 48.5m$ rectangle, containing 18 rooms in the same floor. Figure 4 shows the true radio maps, including all $n = 3318$ RPs.

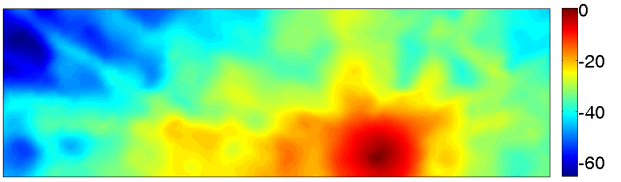


FIGURE 4. True radio map created by WinProp.

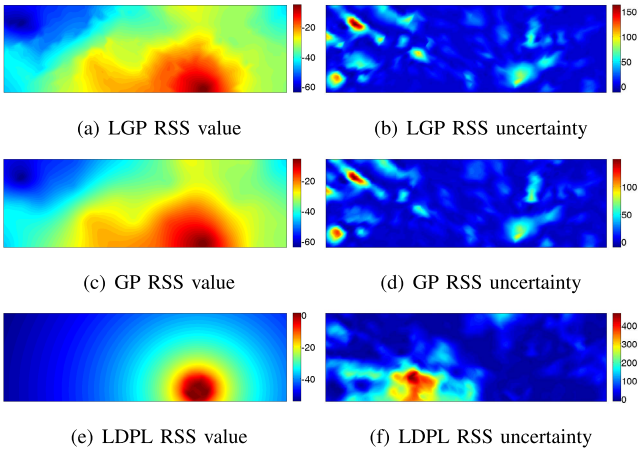


FIGURE 5. The uncertainty on the RSS value(a) radio map using LGP (b) LGP RSS uncertainty (c) radio map using GP, $K = 4$ (d) GP RSS uncertainty (e) radio map using LDPL. (b) LDPL RSS uncertainty.

Figure 5 shows the radio maps created by GP, LGP and LDPL, respectively. In this simulation, we randomly select 80 nodes from the database as training data. And use different algorithms to estimate the other nodes' RSS values. The parameters of the GP are estimated based on the training data, considering all $n = 80$ RPs. In LGP algorithm, L is set to 4. And the Log-Distance Path Model (LDPL) [24], where the parameters of the LDPL model are estimated based on the training data. The uncertainty of RP i is defined as follows:

$$Diff_i = (RSS_i - \widehat{RSS}_i)^2 \tag{6}$$

where $\widehat{RSS}_{i,j}$ and $RSS_{i,j}$ are estimated and true RSS values at RP j, respectively.

As can be observed, the radio maps for GP and LGP are similar to the true radio map. The LDPL model, which is known to fail at positions far from the signal source, resembles less the true radio map.

We also compute the average uncertainty over all positions. The average uncertainty is defined as:

$$Diff = \frac{\sum_{i=1}^{m} Diff_i}{m} \tag{7}$$

GP has the lowest average uncertainty, which is about 8.00, followed by LGP with an average of 9.42. The highest average comes from LDPL, which is 34.86.

For more accurate result, we use $\hat{\sigma}_r$, $RMSE_r$ and *time* to evaluate these three algorithms, $\hat{\sigma}_r$ and $RMSE_r$ are

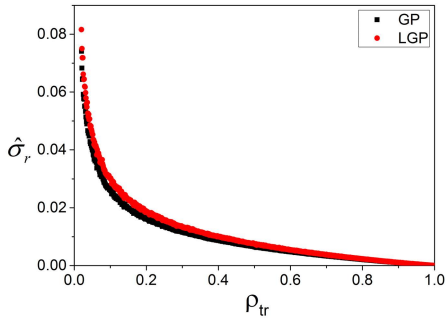


FIGURE 6. The $\hat{\sigma}_r$ of GP and LGP.

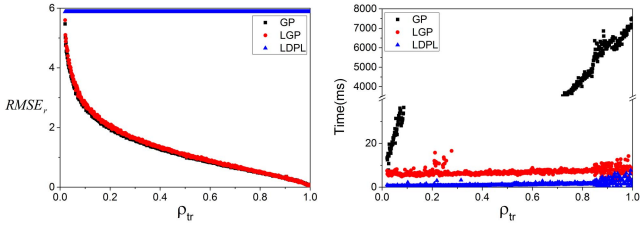


FIGURE 7. $RMSE_r$ and Time complexity vary with different ρ_{tr}. (a) is $RMSE_r$ and (b) is the time spend in creating the virtual database using different algorithms.

defined as:

$$\hat{\sigma}_r^2 = \sum_i \sigma_i^2 / m \tag{8}$$

$$RMSE_r = \sqrt{\frac{\sum_i Diff_i}{m}} \tag{9}$$

where m is the number of virtual RPs.

time refers to the time for calculating all the RPs' RSS values in the database.

And we also define a parameter ρ_* to indicate the density of the database:

$$\rho_* = n_*/S \tag{10}$$

where n_* is the number of RPs in the database. ρ_{tr} and ρ_v refers the density of the training database and virtual database, respectively.

In the following simulation, we set ρ_{tr} varies from 0.02 to 1, the training RPs are selected randomly. For each ρ_{tr}, we simulate 2000 times with $\rho_v = 1$, $\sigma_n^2 = 0$ and $L = 4$. Fig.6 is the result.

In Fig.6, We can see that GP has a smaller $\hat{\sigma}_r$ than LGP, but the differences between them is slightly small. We compare GP, LGP and LDPL in RMSE and the time complexity in Fig.7.

In Fig.7(a), GP performs the best, followed by LGP, and LDPL performs the worst. But the differences between GP and LGP is small. In Fig.7(b), LDPL has the lowest time complexity, followed by LGP, and GP. In summary, LGP keeps a good balance between RMSE and time complexity.

D. THE WEIGHTED K NEAREST NEIGHBORS ALGORITHM

Using the two previous algorithms, we are able to generate the hierarchical database required for our HPA positioning algorithm. With this hierarchical database, we will successively estimate the position at the different levels, starting at the highest level. At each level, we use the weighted K nearest neighbors (WKNN) algorithm, which searches in the database for the K RPs with RSS values closest to the user, and takes a weighted sum of the positions of these nearest RPs to estimate the position. However, as the uncertainty on the RSS value of some RPs in the database can be high, the accuracy of the standard WKNN algorithm can be low. In order to improve the accuracy, we change the algorithm to take into account the uncertainty on the RSS values in the database.

Let us assume we measure the RSS values N times. Assume the user measures the RSS values $\{RSS_{\ell,t}|\ell = 1,\ldots,a\}$ during the t^{th} measurement, where a is the number of signal sources. The signal distance to RP j in the database at level k is defined as

$$d_{j,t} = \sum_{\ell=1}^{a} \delta_{j,\ell}^{(k)} |RSS_{\ell,t} - RSS_{j,\ell}^{(k)}| \tag{11}$$

where $RSS_{j,\ell}^{(k)}$ is the RSS value at RP j for signal source ℓ, and $\delta_{j,\ell}^{(k)}$ is the uncertainty on that RSS value. This distance measure favors more reliable RPs having a small $\delta_{j,\ell}^{(k)}, \ell = 1,\ldots,a$ and will disfavor unreliable RPs with high uncertainty $\delta_{j,\ell}^{(k)}$. Taking into account that in general the uncertainty of a training RP will be lower than that of a virtual RP, the training data is more likely to be selected with this altered algorithm, improving the accuracy.

After the selection of the nearest neighbors, the WKNN algorithm estimates the coordinates of the user by weighting the coordinates of the K selected RPs:

$$(\hat{x}_t, \hat{y}_t) = (\sum_{j=0}^{K} \tilde{w}_{j,t} x_j^{(k)}, \sum_{j=1}^{K} \tilde{w}_{j,t} y_j^{(k)}) \tag{12}$$

where $(x_j^{(k)}, y_j^{(k)})$ are the coordinates of RP j, and $\tilde{w}_{j,t}$ is the normalized weight:

$$\tilde{w}_{j,t} = \frac{1/d_{j,t}^p}{\sum_{i=0}^{K} 1/d_{i,t}^p}. \tag{13}$$

The variance σ is defined as follows:

$$\sigma = \sum_{t=1}^{N} \frac{\sum_{i=1}^{K} \tilde{w}_i[(x_i^{(k)} - \hat{x}_t)^2 + (y_i^{(k)} - \hat{y}_t)^2]}{1 - \sum_{i=1}^{K} \tilde{w}_i^2} \tag{14}$$

If we want more accurate result, we can continue our estimation using lower level sub-database $DB^{(k+1)}$, which contains more RPs. For reducing the time for searching in database to find the most likely match RPs. We can use the previous result to reduce the searching space.

In the previous estimation, we calculate the location based on K nearest RPs in $DB^{(k+1)}$. We put these K RPs in set $TS^{(k)}$.

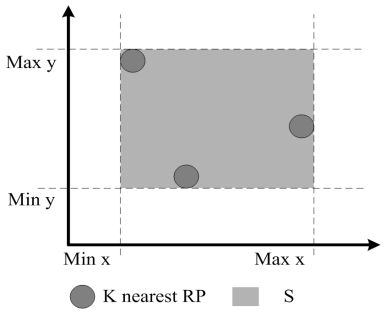


FIGURE 8. Creating *S* based on *TS*.

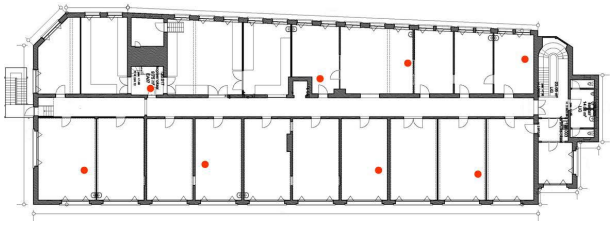


FIGURE 9. Floor plan of the office corridor and the position of the APs.

Based on $TS^{(k)}$, we can estimate the user's most possible located area denoted by $S_t^{(k)}$:

$$S_t^{(k)} = \{(x,y)|x \subset [min\{x_j^{(k)}\}, max\{x_j^{(k)}\}], \\ y \subset [min\{y_j^{(k)}\}, max\{y_j^{(k)}\}]\}, (x_j^{(k)}, y_j^{(k)}) \subset TS^{(k)} \tag{15}$$

Fig. 8 shows how the area $S_t^{(k)}$ is created, where K is set to 3:

The area S_t is the user's most likely located place. We select all the node in $S_t^{(k)}$ from $DB^{(k+1)}$ for fine estimation using the algorithm illustrated in this section.

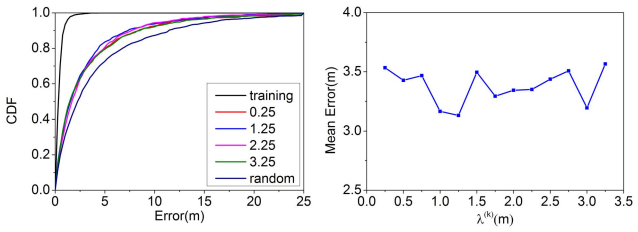
III. NUMERICAL RESULTS

In this experiments, we apply the WiFi fingerprint to test our algorithm. We first compare our proposed virtual database creation algorithm LGP with GP and the widely used Log-Distance Path Loss model. We also evaluate our positioning algorithm with both high and low density training database.

A. SET-UP OF INDOOR EXPERIMENT

Our test environment consists of 18 room in one floor, with the area of $800m^2$. And 8 APs are placed in this area. The floor plan of the corridor and the position of the APs are shown in Fig. 9.

We build a WiFi fingerprint radio map for our environment by means of 3D ray tracing. We use WinProp from AWE communications to create the signal fingerprint database. The database contains 3318 RPs, denoted as DB. We consider the radio map as the ground truth. In the following simulation, we only use 800 random distributed RPs for training. The others are used for testing.



(a) CDF (b) Mean error

FIGURE 10. CDF and Mean error vary with different $\lambda^{(k)}$. The training curve means using the training database for positioning, and random means using 80 randomly selected RPs for positioning.

For evaluating different algorithms, we define *tc* and *RMSE*. *RMSE* is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}[(x_t - \hat{x}_t)^2 + (y_t - \hat{y}_t)^2]}{N}} \tag{16}$$

where (x_t, y_t), $(\hat{x}_t, \hat{x}_t)$ are the true and estimated coordinates, respectively. N is the number of positioning instance.

tc refers to the mean time for locating one single RP.

B. EVALUATING THE PARAMETERS

There are some parameters in HPA, including $\lambda^{(k)}$ in MDA, K in WKNN, $\varepsilon^{(k)}$ and L in LGP. All of these parameters determines the complexity and positioning accuracy of HPA. We first evaluate these parameters to get the optimized values.

1) $\lambda^{(k)}$

$\lambda^{(k)}$ determines the density of $DB_v^{(k)}$. Different $DB_v^{(k)}$ result in different $DB^{(k)}$, as a result, the positioning accuracy might be affected.

In this simulation, we test different $\lambda^{(k)}$. We apply the LGP for estimating the virtual RPs' RSS values. The stardand WKNN algorithm is applied for positioning. The other parameters are set as follows: $\varepsilon = 0$, $K = 4$, $L = 4$,$n^k = 80$; σ_f^2, σ_n^2 and l are estimated using hyper-parameter estimation. We build different $DB^{(k)}$ based on different $DB_v^{(k)}$ for positioning. Results comes from 20000 positioning instance.

Fig. 10 is the positioning result from different $\lambda^{(k)}$:

In this simulation, $\lambda^{(k)}$ is set to 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3, and 3.25, respectively. We also use the training database, containing 800 training RPs, and randomly selected 80 RPs for positioning. Fig.10(a) only contains the result when $\lambda^{(k)}$ is set to 0.25, 1.25, 2.25, and 3.25. In Fig.10, positioning using the training database performs the best, while the random database worst. The difference between variant $\lambda^{(k)}$ is slightly small. In Fig.10(b), $\lambda^{(k)}$ has almost no effect on the performance. But building the database with different $\lambda^{(k)}$ cost different time. Fig.11 gives the time for constructing the virtual database with different $\lambda^{(k)}$:

Fig.11 shows that the time decrease when $\lambda^{(k)}$ increase. The result from Fig.10 and Fig.11 tell us that we can use as large $\lambda^{(k)}$ as possible to reduce the time for constructing the database.

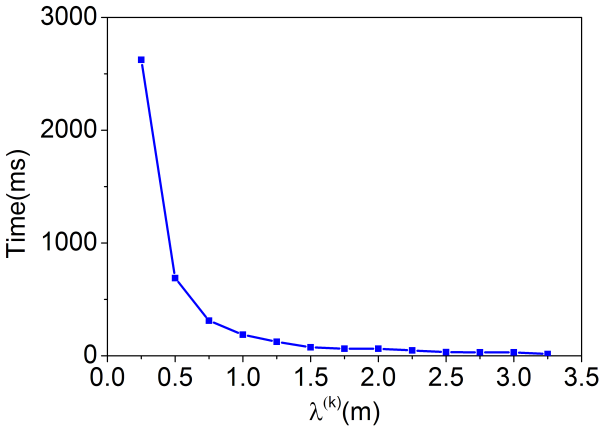


FIGURE 11. Time for building the virtual database.

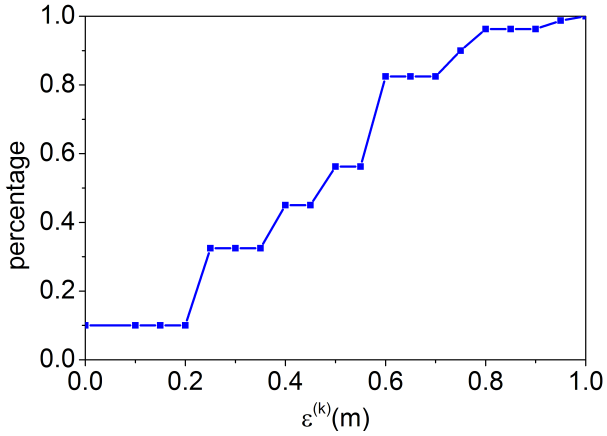


FIGURE 12. percentage of training data in $DB^{(k)}$ varies with $\varepsilon^{(k)}$.

2) $\varepsilon^{(k)}$

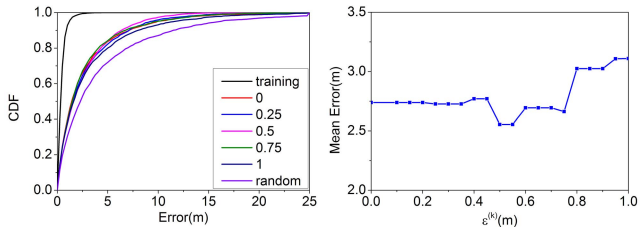
$\varepsilon^{(k)}$ determines the distance that we can use the training data directly. A big $\varepsilon^{(k)}$ might break the distribution of the RPs, but will introduce more reliable RSS values. In this section, we will evaluate the performance with different $\varepsilon^{(k)}$.

Similar with the previous setting, we apply the LGP for estimating the virtual RPs' RSS values, and the traditional WKNN algorithm for positioning. The other parameters are set as follows: $\lambda^{(k)} = 1.25$, $K = 4$, $L = 4$,$n^k = 80$; σ_f^2, σ_n^2 and l are estimated using hyper-parameter estimation. We build $DB^{(k)}$ with different $\varepsilon^{(k)}$ based on $DB_v^{(k)}$ for positioning.

We first look at the percentage of training data in $DB^{(k)}$. A large $\varepsilon^{(k)}$ result in more training data be used. Fig.12 shows the result. Results comes from 20000 positioning instance.

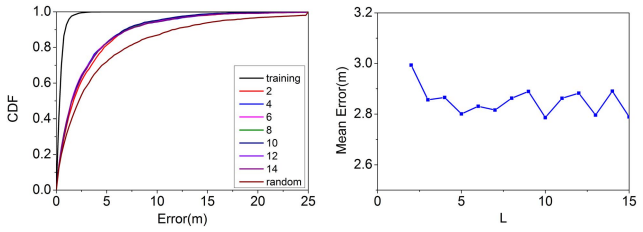
The result in Fig.12 shows what we expected. The percentage increase as the grows of $\varepsilon^{(k)}$.

More training data introduces more reliable data, but the breaks the distribution of the RPs in $DB^{(k)}$. We explore the positioning performance with different $\varepsilon^{(k)}$. Fig.13 gives the result.



(a) CDF (b) Mean error

FIGURE 13. CDF and Mean error vary with different $\varepsilon^{(k)}$. The training curve means using the training database for positioning, and random means using 80 randomly selected RPs for positioning.



(a) CDF (b) Mean error

FIGURE 14. CDF and Mean error vary with different *L*. The training curve means using the training database for positioning, and random means using 80 randomly selected RPs for positioning.

In this simulation, $\varepsilon^{(k)}$ is set to 0, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, and 1, respectively. We also use the training database, containing 800 training RPs, and a randomly selected 80 RPs for positioning. Fig.13(a)only contains the result when $\varepsilon^{(k)}$ is set to 0, 0.25, 0.5, 0.75, and 1. In Fig.13, positioning using the training database performs the best, different $\varepsilon^{(k)}$ doesn't result in significant difference in performance. When $\varepsilon^{(k)} = 1$, which means all the RPs are training nodes, the performance is not the best. When $\varepsilon^{(k)} = 0$, which means the least training RPs are included, the performance is not the worst. Fig.13(b) shows that the best result is achieved when the $\varepsilon^{(k)}$ is set to a middle value.

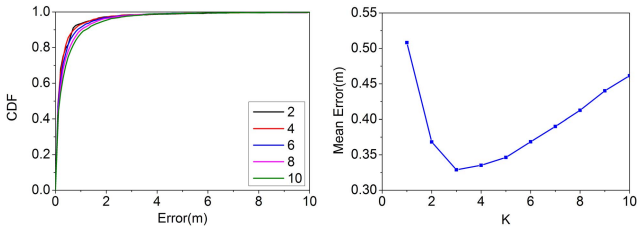
3) *L*

L is the number of training RPs used for estimating the RSS values for a given virtual nodes in LGP. A large L introduces more training data, and more accurate result obtained. But the time for estimating the RSS values will be increased. In this section, we explore to find a good balance between the accuracy and time for building the virtual database.

Similar with the previous setting, we apply the LGP for estimating the virtual RPs' RSS values, and the traditional WKNN algorithm for positioning. The other parameters are set as follows: $\lambda^{(k)} = 1.25$, $K = 4$, $n^k = 80$, $\varepsilon^{(k)} = 0.5$; σ_f^2, σ_n^2 and l are estimated using hyper-parameter estimation. We build $DB^{(k)}$ based on $DB_v^{(k)}$ with different L for positioning. Results comes from 20000 positioning instance.

Fig.14 shows the result.

In this simulation, L is set from 2 to 15. Positioning using training database and randomly selected database are included. In Fig. 14(a), we can see that different $L's$ CDF curve looks similar to each other. When $L = 2$, we find that the positioning accuracy is not as good as others. Fig.14(b) proves our observation. The reason is that using a small L means fewer training data used for estimating the virtual RP's RSS values. As a result, the positioning result is not as good as using more training data. But a large L does not leads to better performance. This result proves assumption 1. We needn't to using all the training data for estimation.



(a) CDF (b) Mean error

FIGURE 15. CDF and Mean error vary with different *K*.

4) *K*

K is the number of nodes used for positioning. In this simulation, we want to find the best K for estimation. We apply the traditional WKNN algorithm for positioning based on the training database. We set K increase from 1 to 10. Results comes from 20000 positioning instance.

Fig.15 shows the result.

The result from Fig.15(a) shows that a smaller or larger K is not good choice. Fig.15(b) tells us that using 3 nodes for positioning performs the best.

C. EVALUATING THE IMPROVED WKNN ALGORITHM

The difference between standard WKNN and improved WKNN algorithm is the distance definition. In this simulation, we want to evaluate the improved WKNN algorithm both in sparse and dense database. We create two virtual databases using MDA and LGP based on 40 randomly selected RPs. One is sparse virtual database, containing 40 virtual points denoted as $DB^{(1)}$, while the other one contains 400 virtual points, denoted as $DB^{(2)}$. In MDA algorithm, $\lambda^{(1)} = \lambda^{(2)} = 1.25$. In LGP algorithm, $L = 5$, $\varepsilon^{(1)} = \varepsilon^{(2)} = 0.5$. In standard and improved WKNN algorithm, $K = 3$.

Fig.16 shows the CDF and Mean Error for these two algorithms using different databases.

From Fig.16, we can see that the improved algorithm perform better when we use the sparse database. In sparse database, the average distance from the testing node to the nearby RPs is large than that in dense nodes. But the number of training RPs is limited, and we set $\varepsilon^{(1)} = \varepsilon^{(2)} = 0.5$. As a result, we have more possibility to select the more reliable data in sparse database.

When we use the sparse database for positioning, the mean error for the standard algorithm is 2.2479m, while the

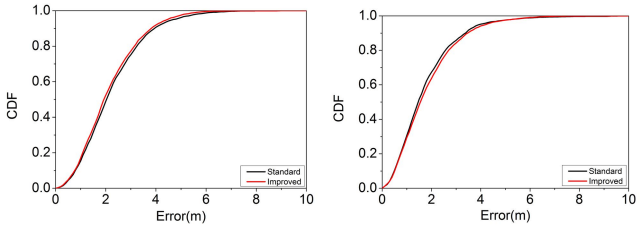


FIGURE 16. Evaluating improved WKNN using different database. (a) is using a database containing 40 virtual RPs. (b) is using a database containing 400 virtual RPs.

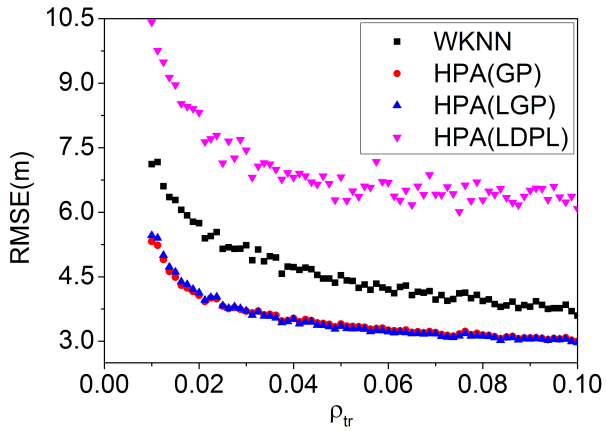


FIGURE 17. RMSE of WKNN and HPA using sparse training database.

improved algorithm is 2.1471m. The positioning accuracy has been improved by the proposed algorithm. When we use the dense database, the positioning accuracy drops down as illustrated in Fig. 16(b).

D. EVALUATING HPA

1) USING SPARSE TRAINING DATABASE

Training and updating a low density of training database is much easier than a dense one. Here we evaluate HPA using the low density of training database.

In the Simulation, we set ρ_{tr} increases from 0.01(8 RPs in 800 m^2 area) to 0.1(80 RPs in 800 m^2 area), and the RPs in ρ_{tr} are selected randomly from DB. In HPA, we create the database with only two levels using GP, LGP and LDPL, respectivily. $\rho^{(1)} = 0.05$, $\rho^{(2)} = 0.5$, $\lambda^{(1)} = \lambda^{(2)} = 1.25$, $\varepsilon^{(1)} = \varepsilon^{(2)} = 0.5$. In LGP, $L = 5$. In the standard WKNN, we use DB_{tr} for positioning. And we set $K = 3$ both in standard and improved WKNN. For each ρ_{tr}, we simulate 800 times. In each simulation, we test 2000 positioning instances. In each positioning instances, we use DB to generate signal strength measurement by adding Gaussian noise $N(0, 5)$. We compare their RMSE, which defined in Equation (16), and average positioning time for one node.

Fig. 17 gives the simulation results.

In Fig. 17, there are four curves. WKNN means using the standard WKNN algorithm, and the database is the

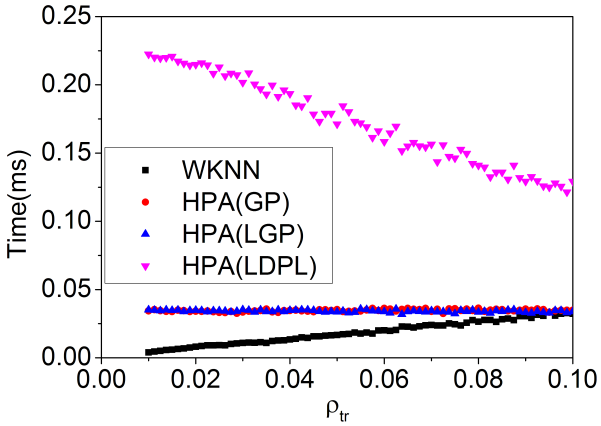


FIGURE 18. Time of WKNN and HPA using sparse training database.

training database. The average RMSE is 4.59. HPA(GP), HPA(LGP), and HPA(LDPL) mean using the HPA algorithm for positioning, and the virtual databases are built using GP, LGP, and LDPL algorithm, respectively. As we can observed from this Figure, HPA(LDPL) performs the worst. The average RMSE is 7.05. HPA(GP) and HPA(LGP) perform nearly the same, the average RMSE is 3.49 and 3.48, respectively. Compared with standard WKNN, the proposed algorithm improves the RMSE for about 24.2%. Fig.18 gives the time for locating one node.

In Fig.18, we find that WKNN cost least time, this is because there are less RPs for positioning. HPA(LDPL) cost the most time, this is because in the first stage, the possible region is very large. As a result, in the second stage, the algorithm has to search in a large database. HPA(GP) and HPA(LGP) cost nearly the same. When the training database's density is about 0.1, WKNN, HPA(GP) and HPA(LGP) cost the same time. But in Fig.17, we can see that HPA(GP) and HPA(LDPL) have less RMSE. And in Fig.7(b), we see that GP cost far more time for building the virtual database. These result means the proposed algorithm performs the best when the training database is sparse.

2) USING DENSE TRAINING DATABASE

Assuming we have a dense training database DB_{tr}. For comparing WKNN and HPA, we assumes ρ_{tr} varies from 0.1 to 1. The RPs in the training database are selected randomly from DB. All the other parameters are set the same as the previous section. Fig. 19 gives the simulation results.

In Fig. 19, all the four curves mean the same with Fig.17, but here we use more dense training database. As illustrated in Fig.19, HPA(LDPL) performs the worst, the average RMSE is 4.68. Followed by the standard WKNN algorithm, while the average RMSE is 3.22. HPA(GP) and HPA(LGP) performs nearly the same, the average RMSE is 2.874 and 2.873, respectively. The performance has been improved for about 10.8%. Fig.20 is the time for locating one node.

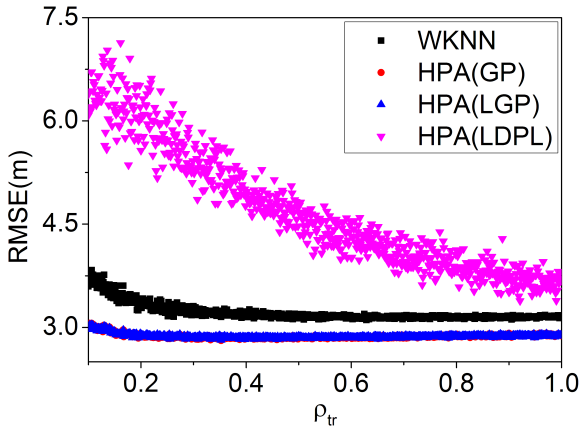


FIGURE 19. RMSE of WKNN and HPA using dense training database.

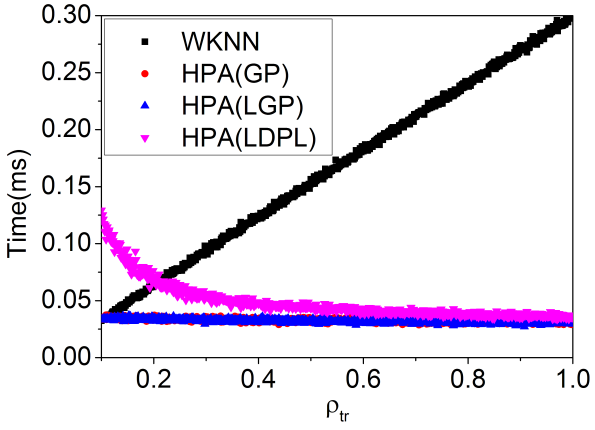


FIGURE 20. Time of WKNN and HPA using dense training database.

In Fig.20, the time for WKNN increase with ρ_{tr}, this is because the algorithm has to search in a increasing scale database. But the time for HPA(LDPL) decrease with ρ_{tr}, this is because more training RPs are introduced as the increase of ρ_{tr}, where Fig.12 gives the result. HPA(GP) and HPA(LGP) cost the same. But Fig.7(b) tells us that GP cost far more time for building the virtual database than LGP. These result also means the proposed algorithm performs the best when the training database is dense.

In this subsection, we prove that the proposed algorithm performs better both using sparse and dense training database.

IV. CONCLUSION AND FUTURE WORKS

Wireless fingerprint technique has the advantage of low deployment cost, supply for reasonable accuracy and easily to be applied to mobile devices. As a result, fingerprinting has attracted a lot of attentions. The positioning accuracy of fingerprint technique is highly dependent on the density of RPs in the fingerprint database. However, constructing a fingerprint database, with high density fingerprint samples, is labor-intensive or impossible in some cases. And even if it was possible to get a high density of fingerprint database, data retrieve from the large database is time consuming, and the database has to be updated as the environment changes.

For these problems in fingerprinting, we introduce the DLOD(Discrete Level of Detail) from computer graphics to the fingerprint localization problem to propose HPA. This algorithm can be applied to the both high and low density of training database.

In HPA, we first propose MDA to transform the different density of training database into uniformly distributed databases, which contain different density of sub-databases. And then, we propose LGP to estimate the RSS values for all the virtual RPs. The higher level built with low density of fingerprints gives coarse estimation, while the lower level contains dense data to provide fine estimation. There are two advantages in HPA. Firstly, the time for positioning is reduced. This is because, in HPA, we only search the matched reference points in a small database, instead of a large database. Secondly, the positioning accuracy is increased. This is because LGP will supply more reference points for positioning.

Simulation results show that when we have a dense training database, the positioning accuracy can be improved for about 10.8%. And when the training database contains low density fingerprints, the positioning accurate can be improved for about 24.2%. The results also imply us that we can select the RPs for training database based on MDA.

This algorithm can be applied to both high and low density training database, need no more specialized hardware, and can reuse the existing Wi-Fi infrastructure and fingerprint database, implying it's robust for daily used.

Our future work is concentrate on building a larger experiment environment to test our algorithm, and more levels will be tested.

REFERENCES

[1] A. Pérez-Navarro, J. Torres-Sospedra, R. Montoliu, J. Conesa, R. Berkvens, G. Caso, C. Costa, N. Dorigatti, N. Hernández, S. Knauth, E. S. Lohan, J. Machaj, A. Moreira, and P. Wilk, "Challenges of fingerprinting in indoor positioning and navigation," in *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation*. Amsterdam, The Netherlands: Elsevier, 2019, pp. 1–20.

[2] A. Popleteev, "Improving ambient FM indoor localization using multipath-induced amplitude modulation effect: A year-long experiment," *Pervasive Mobile Comput.*, vol. 58, Aug. 2019, Art. no. 101022.

[3] A. Canepa, Z. Talebpour, and A. Martinoli, "Automatic calibration of ultra wide band tracking systems using a mobile robot: A person localization case-study," in *Proc. Int. Conf. Indoor Positioning Indoor Navigation (IPIN)*, Sep. 2017, pp. 1–8.

[4] C.-I. Chesneau, M. Hillion, J.-F. Hullo, G. Thibault, and C. Prieur, "Improving magneto-inertial attitude and position estimation by means of a magnetic heading observer," in *Proc. Int. Conf. Indoor Positioning Indoor Navigation (IPIN)*, Sep. 2017, pp. 1–8.

[5] Google Maps. (2005). *Google Maps Indoor*. [Online]. Available: http://www.google.cn/maps/about/partners/indoormaps/

[6] B. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," in *Proc. IJCAI*, vol. 7, 2007, pp. 2480–2485.

[7] (2005). *RTMaps*. [Online]. Available: http://www.rtmap.com/

[8] Q. Chang, Q. Li, Z. Shi, W. Chen, and W. Wang, "Scalable indoor localization via mobile crowdsourcing and Gaussian process," *Sensors*, vol. 16, no. 3, p. 381, Mar. 2016.

[9] H. Koyuncu and S.-H. Yang, "Indoor positioning with virtual fingerprint mapping by using linear and exponential taper functions," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 1052–1057.
[10] C. Serôdio, L. Coutinho, L. Reigoto, J. Matias, A. Correia, and P. Mestre, "A lightweight indoor localization model based on motley-keenan and cost," in *Proc. World Congr. Eng.*, vol. 2, 2012, pp. 1323–1328.
[11] S. Jung, C.-O. Lee, and D. Han, "Wi-Fi fingerprint-based approaches following log-distance path loss model for indoor positioning," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Aug. 2011, pp. 1–2.
[12] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, Nov. 2005.
[13] B. F. D. Ferris, D. Häehnel, and D. Fox, "Gaussian processes for signal strength-based location estimation," in *Proc. Robot., Sci. Syst.*, Aug. 2006, pp. 1–8.
[14] X. Niu, C. Zhang, A. Wang, J. Liu, and Z. Wang, "A crowdsourcing-based Wi-Fi fingerprinting mechanism using un-supervised learning," in *Proc. Int. Conf. Wireless Algorithms, Systems, Appl.*, 2018, pp. 357–373.
[15] P. Maher and R. A. Malaney, "A novel fingerprint location method using ray-tracing," in *Proc. IEEE Global Telecommun. Conf.*, Nov. 2009, pp. 1–5.
[16] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization," *IEEE Trans. Robot. Autom.*, vol. 17, no. 2, pp. 125–137, Apr. 2001.
[17] M. Nowicki and J. Wietrzykowski, "Low-effort place recognition with WiFi fingerprints using deep learning," in *Proc. Int. Conf. Automat.*, Mar. 2017, pp. 575–584.
[18] K. S. Kim, S. Lee, and K. Huang, "A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting," *Big Data Anal.*, vol. 3, no. 1, p. 4, Dec. 2018.
[19] K. S. Kim, R. Wang, Z. Zhong, Z. Tan, H. Song, J. Cha, and S. Lee, "Large-scale location-aware services in access: Hierarchical building/floor classification and location estimation using Wi-Fi fingerprinting based on deep neural networks," *Fiber Integr. Opt.*, vol. 37, no. 5, pp. 277–289, 2017.
[20] K. S. Kim, "Hybrid building/floor classification and location coordinates regression using a single-input and multi-output deep neural network for large-scale indoor localization based on Wi-Fi fingerprinting," in *Proc. 6th Int. Symp. Comput. Netw. Workshops (CANDARW)*, Nov. 2018, pp. 196–201.
[21] J. H. Clark, "Hierarchical geometric models for visible surface algorithms," *Commun. ACM*, vol. 19, no. 10, pp. 547–554, Oct. 1976.
[22] J. Ribelles, A. López, and O. Belmonte, "An improved discrete level of detail model through an incremental representation," in *Proc. Theory Pract. Comput. Graph.*, Sheffield, U.K., 2010, pp. 59–66.
[23] AWE. (2005). *AWE Communications Homepage Available From*. [Online]. Available: http://www.awe-communications.com/
[24] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2010, pp. 173–184.

QIANG CHANG (M'11–SM'16) received the M.Sc. and Ph.D. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2011 and 2016, respectively. He is currently an Assistant Research Fellow with the National Innovation Institute of Defense Technology (NIIDT). His research interests include indoor positioning and communication networks.

SAMUEL VAN DE VELDE (M'10–SM'16) received the M.Sc. degree in applied sciences from Ghent University, Gent, Belgium, in 2010. He is currently the Founder and a CTO of Pozyx accurate positioning.

HEIDI STEENDAM (M'01–SM'06) received the M.Sc. degree in electrical engineering and the Ph.D. degree in applied sciences from Ghent University, Gent, Belgium, in 1995 and 2000, respectively. Since September 1995, she has been with the Digital Communications Research Group, Department of Telecommunications and Information Processing, Faculty of Engineering, Ghent University, first in the framework of various research projects, and since October 2002, has been a full-time Professor of digital communications. She has authored more than 140 scientific articles in international journals and conference proceedings, for which she received several best paper awards. Her current research interests include statistical communication theory, carrier and symbol synchronization, bandwidth-efficient modulation and coding, cognitive radio, and cooperative networks and positioning. Since 2002, she has been an Executive Committee Member of the IEEE Communications and Vehicular Technology Society Joint Chapter, Benelux Section. Since 2012, she has been the Vice-Chair. She has been active in various international conferences as a Technical Program Committee Chair/Member and a Session Chair. In 2004 and 2011, she was the Conference Chair of the IEEE Symposium on Communications and Vehicular Technology in the Benelux. She is currently an Associate Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS and the *EURASIP Journal on Wireless Communications and Networking*.