sensors

MDPI

Article

Scalable Indoor Localization via Mobile Crowdsourcing and Gaussian Process

Qiang Chang, Qun Li *, Zesen Shi, Wei Chen and Weiping Wang

College of Information Systems and Management, National University of Defense Technology, Changsha 410073, China; changqiang@nudt.edu.cn (Q.C.); nudtshizesen@126.com (Z.S.); weichen@nudt.edu.cn (W.C.); wang.wp2010@gmail.com (W.W.)
* Correspondence: liqun@nudt.edu.cn; Tel.: +86-137-8708-2801

Academic Editors: Lyudmila Mihaylova and Byung-Gyu Kim
Received: 21 January 2016; Accepted: 14 March 2016; Published: 16 March 2016

Abstract: Indoor localization using Received Signal Strength Indication (RSSI) fingerprinting has been extensively studied for decades. The positioning accuracy is highly dependent on the density of the signal database. In areas without calibration data, however, this algorithm breaks down. Building and updating a dense signal database is labor intensive, expensive, and even impossible in some areas. Researchers are continually searching for better algorithms to create and update dense databases more efficiently. In this paper, we propose a scalable indoor positioning algorithm that works both in surveyed and unsurveyed areas. We first propose Minimum Inverse Distance (MID) algorithm to build a virtual database with uniformly distributed virtual Reference Points (RP). The area covered by the virtual RPs can be larger than the surveyed area. A Local Gaussian Process (LGP) is then applied to estimate the virtual RPs' RSSI values based on the crowdsourced training data. Finally, we improve the Bayesian algorithm to estimate the user's location using the virtual database. All the parameters are optimized by simulations, and the new algorithm is tested on real-case scenarios. The results show that the new algorithm improves the accuracy by 25.5% in the surveyed area, with an average positioning error below 2.2 m for 80% of the cases. Moreover, the proposed algorithm can localize the users in the neighboring unsurveyed area.

Keywords: WLAN; indoor localization; radio map; mobile crowdsourcing; gaussian process; Bayesian algorithm

1. Introduction

The difficulty of determining the location of mobile users within buildings has been extensively studied for decades, due to potential applications in the mobile networking environment [1]. With the wide availability of 802.11 WLAN networks, wireless localization using Received Signal Strength Indication (RSSI) fingerprinting [2] has attracted a lot of attention.

Fingerprint indoor positioning consists of two phases: training and localization [3]. During the training phase, a database of location-fingerprint mapping is constructed. In the localization phase, the users send location queries with the current RSS fingerprints to the location server; the server then retrieves the signal database and returns the matched locations.

The accuracy of fingerprinting techniques is highly dependent on the density of the signal database. Building and maintaining a high-density database are not easy, however, for two reasons.

Firstly, building a high-density fingerprint database is labor intensive, expensive, and even impossible in some cases. Taking a 50 m $\times$ 50 m floor as an example, if we want to build a fingerprint database with a 1 m sample distance, we would have to collect 2500 samples. For each sample, we need to measure several times to get reliable results. Sometime it is impossible to collect signal fingerprints from certain locations, because of the complex local environment.

Secondly, maintaining a large signal database is expensive. As the environment changes over time due to furniture or signal sources being moved, the fingerprints diverge from those in the database. This means that the entire area needs to be re-surveyed in order to update the database. As indoor environments often change, the database would require frequent updates, which would be time-consuming and expensive.

Even though fingerprint indoor localization has many advantages, and some commercial products have been developed on this technology, such as Google Maps [4], WiFiSlam [5], and so on, challenges still exist in its application. In area without calibration data, however, this algorithm breaks down. Real-world deployment of such positioning systems often suffers the problem of sparsely available signal data. For example, Google has collected floor plans for over 10,000 locations. However, only a few of these radio maps are available for positioning.

For these problems in fingerprinting, researchers are continually searching for better algorithms to create and update dense databases more efficiently. The popularization of smartphones makes mobile crowdsourcing fingerprint localization more practical. However, designing a sustainable incentive mechanism of crowdsourcing remains a challenge. In this paper, we propose a scalable indoor positioning algorithm via mobile crowdsourcing and Gaussian Process. The basic idea behind our proposed algorithm is simple: we first propose a Minimum Inverse Distance (MID) algorithm to build a virtual database with uniformly distributed virtual Reference Points (RP). The area covered by the virtual RPs can be larger than the surveyed area. A Local Gaussian Process (LGP) is then applied to estimate the virtual RP's RSSI values based on the crowdsourced training data. Finally, we improve the Bayesian algorithm to estimate the user's location using the virtual database. All the parameters are optimized by simulations, and the new algorithm is tested on real-case scenarios.

In summary, we make the following major contributions:

(1) We propose a MID algorithm to build a virtual database with uniformly distributed virtual RPs. The area covered by the virtual RPs can be larger than the surveyed area.

(2) The Local Gaussian Process (LGP) is applied to estimate the virtual RPs' RSSI values based on the crowdsourced training data.

(3) Bayesian algorithm is improved to estimate the user's location using the virtual database.

(4) We optimize all the parameters in the proposed algorithm by simulations.

(5) An Android app is developed to test the proposed algorithm on real-case scenarios.

The rest of the paper is organized as follows: Section 2 discusses related work on building and maintaining a dense fingerprint database. Section 3 describes the details of the proposed algorithm. Section 4 optimizes the parameters and evaluates the positioning algorithm, and Section 5 concludes the paper.

2. Related Works

For the challenges of fingerprint positioning, much time and effort has been put into building and maintaining a dense fingerprint database with less effort [6]. Except for point by point measurement, there are mainly five ways to construct and maintain a fingerprint database.

The first is crowdsourcing [7–11]. The users are also database constructors. The database is updated with the most recently measured RSS uploaded by the users [12,13]. However, designing a sustainable incentive mechanism of crowdsourcing remains a challenge [14].

The second method is building the database with mathematical models. The most widely used model is the Log-Distance Path Loss (LDPL) [15–18] model. However, the indoor environments are so complex that no simple mathematical model exists to accurately predict the RSS values. Practically, LDPL only gives good results close to the AP.

Ray-Tracing [19–21] is the third method. However, for accurate ray tracing, you need a very detailed description of the environment such that all the reflections that eventually characterize the

received signal can be simulated. Furthermore, this approach is very computationally demanding. Because of these reasons, it is only viable for small setups.

The fourth method is Simultaneous Localization and Mapping (SLAM) [22,23]. In SLAM, the database is populated on the fly, provided that the users are equipped with a receiver and an IMU. In general, the accuracy of positioning with this technique is lower because the database is less accurate.

The fifth way is the combination of the previous methods. A few RPs cover a large range of area and are collected or generated by the previous method. The remaining RP RSS values are estimated mathematically. Linear and exponential taper functions are used by [24]; the Motley–Keenan model [25] and a semi-supervised manifold learning technique [26] are also used by researchers [27]. Liqun Li propose Modellet [28] to approximate the actual radio map by unifying model-based and fingerprint-based approaches. However, their algorithm only works for nodes near the Access Points (APs). Gaussian Process (GP) [29] is a non-parametric model that estimates Gaussian distribution over functions based on the training data [30]. GP is suitable for estimating RSS values [31]. However, GP is computational consuming, meaning it is not a satisfactory method of generating a large scale area's signal strength.

There are also some other researchers that improved fingerprinting performance by introducing new sensors. For example, IMU [32,33], barometer [34], and so on. However, extra running sensors not only consume more battery, but also bring in new errors. Some such algorithms required the sensors to keep running even if the user does not need the positioning service, which is not suitable for our daily use.

Our study is motivated by these pioneer works, but we approached the problem from a different angle and mainly focus on a scalable indoor positioning algorithm that works both in surveyed and unsurveyed areas. We propose a novel algorithm to create WLAN radio map by mobile crowdsourcing and Gaussian Process. We first propose a Minimum Inverse Distance (MID) algorithm to build a virtual database with uniformly distributed virtual Reference Points (RPs). A Local Gaussian Process (LGP) is then applied to estimate the virtual RPs' RSSI values based on the crowdsourced training data. Finally, we improve the Bayesian algorithm to estimate the user's location using the virtual database. We didn't use the crowdsourced data for positioning directly, and we didn't introduce other sensors to improve the performance either.

3. Materials and Methods

3.1. Problem Setting and Algorithm Overview

We only concentrate on the 2D positioning problem in this paper. Assuming the target area is denoted as P, the area of P is $S(m^2)$. There are a Wi-Fi APs in the target area.

In crowdsourced fingerprint positioning, the radio map is created by users. The RSS values in different RPs from different signal sources are measured and uploaded to the database together with the coordinates. The coordinates come from another positioning system, such as GNSS, or specified by the users.

Assuming we have built a signal database DB^{Crowd} by crowdsourcing, and there are n RPs in DB^{Crowd}. The RPs are denoted as $RP_i = \{p_i, F_i, \sigma_i\}, i = 1, 2, \cdots, n$, where $p_i = (x_i, y_i)$ and $F_i = \{(Mac_j, RSS_{i,j}), j = 1, 2, \cdots, a\}$. σ_i is the measurement variance. The density of DB^{Crowd} is denoted as $\rho^{Crowd} = n/S$. The problem in fingerprint localization is estimating the user's current coordinate p_t at time t based on the measurement F_t and the database DB^{Crowd}.

The density of DB^{Crowd} will be different from region to region. As a result, the positioning accuracy will be different for the area. If we want a continuous positioning performance, we need a database with uniformly distributed reference points. There also might be certain areas without RPs, e.g., because of the complex local environment or not covered due to some other reasons. This method breaks down.

In this paper, we propose a novel algorithm to create a virtual WLAN radio map by mobile crowdsourcing and Gaussian Process for scalable indoor positioning. This virtual radio map is denoted as $DB^{(v)}$. $DB^{(v)}$ contains m RPs, so that the density of the virtual database is $\rho^{(v)} = m/S$. The i^{th} RP in $DB_{(v)}$ is $RP_i^{(v)}$. $RP_i^{(v)} = \{p_i^{(v)}, F_i^{(v)}, \sigma_i^{(v)}\}$, where $p_i^{(v)} = (x_i^{(v)}, y_i^{(v)})$ and $F_i^{(v)} = \{(Mac_j, RSS_{i,j}^{(v)}), j = 1, 2, \cdots, a\}$. $RSS_{i,j}^{(v)}$ is AP j's RSSI measured at RP i, and $\sigma_i^{(v)}$ is the variance of the measurement. $RSS_{i,j}^{(v)}$ is estimated using our proposed Local Gaussian Process (LGP) based on DB^{Crowd}. The user makes use of $DB^{(v)}$ for positioning. Figure 1 shows the framework of the proposed algorithm.

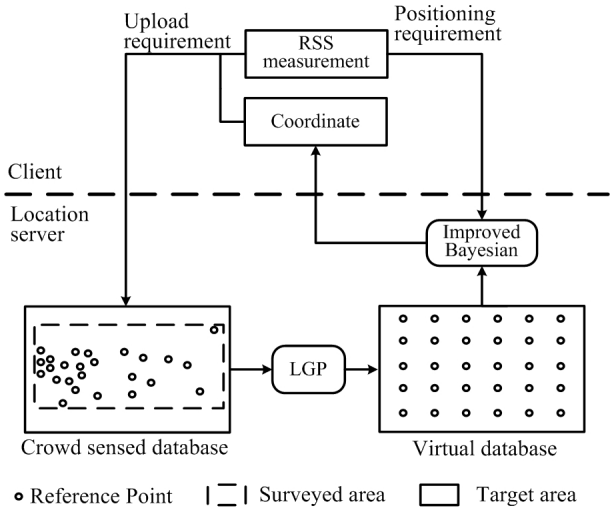


Figure 1. Framework of the proposed algorithm. RSS: Received Signal Strength; LGP: Local Gaussian Process.

After collecting RSS values from surrounding APs, if the user gets the current coordinate by other methods, he can upload the fingerprint containing the coordinate and the RSS values to the server. The server will add the fingerprint to DB^{Crowd}, and update $DB^{(v)}$ using LGP. If the user wants to estimate current location, he can send the positioning requirement, including the RSS measurement, to the server. The server will estimate the coordinate using the proposed Bayesian algorithm based on $DB^{(v)}$ and then send the result to the user.

In the next section, we first built a dense virtual database by introducing uniformly distributed virtual RPs in the area, and then we propose the Local Gaussian Process (LGP) to estimate the virtual RPs' RSSI values and the variance. We improved the Bayesian algorithm to estimate the user's location using the virtual database.

3.2. Building the Dense Virtual Database

As stated earlier, the fingerprints in the virtual database should be selected as uniformly as possible over the target area. m is the number of RPs in $DB^{(v)}$. However, for general values of m, 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 Inverse Distance (MID) algorithm. In this algorithm, the selection of the positions of the RPs is based on a virtual sample database DB^{Sample}, which is constructed by placing a square grid in the target area with grid size λ, where the positions of the virtual RPs are selected as the corners of the squares in the grid. Assuming the target area has size $x_{max} \times y_{max}$, the number of virtual positions equals $\lfloor x_{max}/\lambda \times y_{max}/\lambda \rfloor$. The m positions of the RPs for virtual database $DB^{(v)}$ are selected out of the sample database DB^{Sample}. We initialize the algorithm by randomly choosing one virtual position RP_e from DB^{Sample}:

$DB^{(v)} = \{RP_e\}$. The other $m-1$ positions are picked from the virtual database DB^{Sample} based on the measure in Equation (1):

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

where (x_i, y_i) is the coordinate of the candidate position $\in DB^{Sample}$ and (x_j, y_j) are the coordinates of the RP positions already present in the database $DB^{(v)}$. The virtual position that minimizes Dis_i is selected and added to the database $DB^{(v)}$. Because the measure function Dis_i is inversely proportional to the Euclidean distances between the candidate RP and the RPs in the database $DB^{(v)}$, 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 $DB^{(v)}$ will be distributed uniformly and expand to the very edges of the target area. We call this algorithm as Minimum Inverse Distance (MID) algorithm. Details of MID are shown in Algorithm 1.

Algorithm 1

Require the target area P, the distance λ between neighbor virtual RPs in DB^{sample}
the number m of RPs we want to select.
Ensure select RPs every λ meters in P to build DB^{sample}
Ensure randomly select RP_e from DB^{sample}, $DB^{(v)} = \{RP_e\}$
While($|DB^{(v)}| \neq m$)
For all($RP_i \subset DB^{sample}$)
Calculate Dis_i using Equation (1)
End all
$RP = \arg \min_{RP_i \in DB^{sample}} Dis_i$
$DB^{(v)} \leftarrow RP$
EndWhile

To illustrate MID, we consider an area P of 19.5 m $\times$ 48.5 m and $\lambda = 0.5$ m. Figure 2 shows the positions of the RPs in $DB^{(v)}$ when $m = 100$ and 200 RPs are selected out of the virtual sample database DB^{Sample}. Further, Figure 2 shows the positions of the RPs when the RPs are selected randomly from DB^{Sample}.

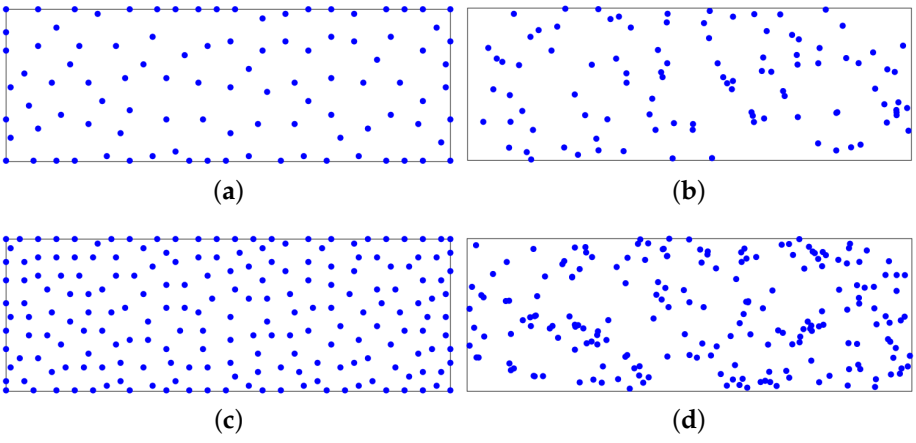


Figure 2. Positions of the Reference Points (RPs) (**a**) Minimum Inverse Distance (MID) , $m = 100$; (**b**) randomly, $m = 100$; (**c**) MID, $m = 200$; (**d**) randomly, $m = 200$.

As can be observed, the proposed algorithm is able to select any number of RPs spatially uniform over the target area.

After the positions of the RPs in database $DB^{(v)}$ are selected with MID, the RSS values and the variance for these RPs need to be determined. To this end, we compare the positions of the RPs in $DB^{(v)}$ with those in DB^{Crowd}. Whenever one or more RPs in DB^{Crowd} are within a distance ε of a RP RP_i in DB^{Crowd}, we will replace the position of the RP in $DB^{(v)}$ with the position of the nearest RP in DB^{Crowd}, together with its RSS values and the variance on the measurement. If no RPs in DB^{Crowd} are within a distance ε of a RP RP_i in DB^{Crowd}, the Local Gaussian Process (LGP) algorithm will be used to estimate the RSS values and their variance in RP_i.

The resulting virtual database $DB^{(v)}$ is determined by three parameters: the number m of RPs in $DB^{(v)}$, the distance λ between RPs in the virtual sample database, and the radius ε within which nearby training RPs are looked for. The number m of RPs is defined by the positioning accuracy. The distance λ determines not only the spatial uniformity of the resulting RPs, but also the complexity of the algorithm: by reducing λ, the RPs will be placed more uniformly over the area P, but the complexity of MID increases as the number of virtual RPs to be searched increases in an inverse proportion to the quad-rate of λ. Finally, the radius ε will also have an influence on the positioning accuracy. When the radius is small, the resulting database $DB^{(v)}$ will have a more uniform placement of RPs, but the probability of finding a nearby training RP decreases, such that the RSS of more RPs needs to be determined using the LGP algorithm. On the other hand, when the selected radius is large, the resulting database $DB^{(v)}$ will be less spatially uniform, but more training RPs will be present in $DB^{(v)}$. In Section 4, we will optimize them before positioning.

3.3. Local Gaussian Process

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 [29]. 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. 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 [5]. 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:

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

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

with σ_n^2 is the measurement variance, $\mathbf{k}_*(i) = k(\mathbf{c}_*, \mathbf{c}_i)$, $i = 1, \ldots, n$. 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 hundred RPs, computing the RSS values with Equations (4) and Equation (5) is computationally demanding because of the inversion of the large covariance matrix Equation (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 values of those far away RPs and the RSS values 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 the 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 Equations (4) and (5) by only considering the L nearest to RPs. That is, $\mathbf{k}_*$ and ρ reduce to a $L \times 1$ vector, and $cov\rho$ Equation (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 Equation (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 [35], denoted as DB. The area is a 19.5 m $\times$ 48.5 m rectangle, containing 18 rooms in the same floor. The true radio map contains 3318 uniformly distributed RPs. We select 100 RPs from DB, which covers part of the target area. Figure 3 shows the true radio maps and the distribution of the selected RPs.

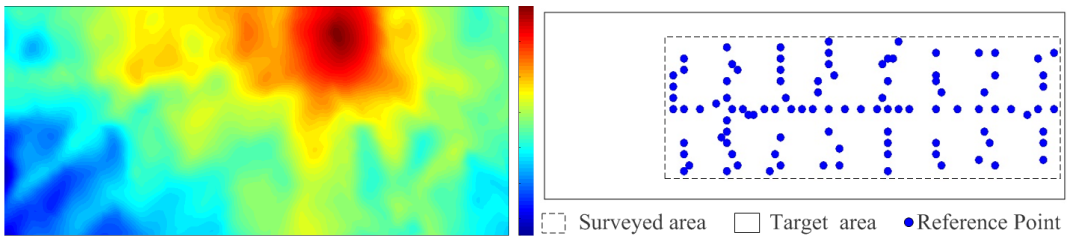


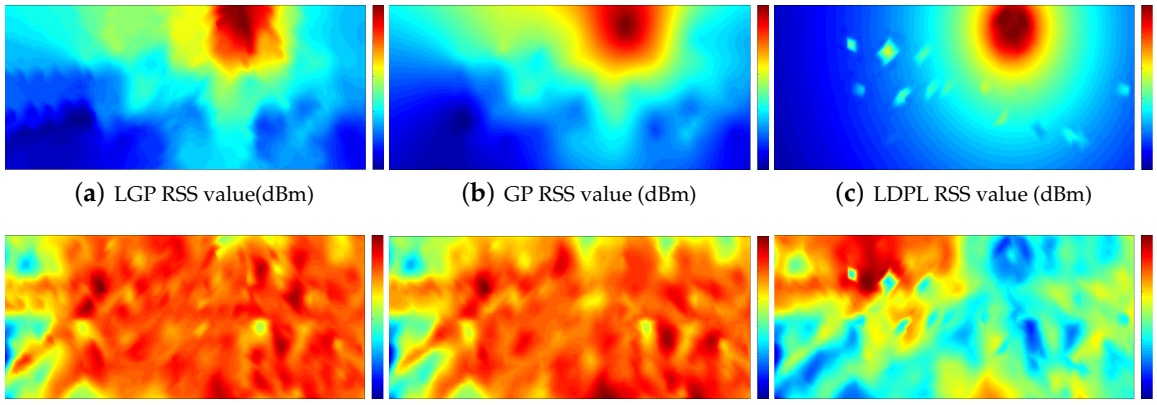
Figure 3. True radio map and the distribution of RPs. (**a**) True Radio map of the whole target area; (**b**) Distribution of Reference Points.

We apply the proposed LGP to create the radio map for the target area. Part of the un-surveyed areas are included.

There are some algorithms can be applied to build the radio map rapidly presented in Section 2, such as crowdsourcing, ray-tracing, SLAM, and mathematical models. We did not supply enough comparison with all of these techniques because different algorithms rely on different equipment and input. It is not straightforward to make comparisons between different algorithms in different conditions. Our study mainly focuses on the mathematical model. As a result, we only make comparisons between the widely used mathematical models, including Gaussian Process (GP) and Log-Distance Path Model (LDPL). Figure 4 is the simulation result. During the simulation, we set $\lambda = 0.5$, $L = 4$, $m = 800$, and $\varepsilon = 0.5$. In the Log-Distance Path Model (LDPL) [36], where the parameters of the LDPL model are estimated based on the training data, the uncertainty of RP i is defined as follows:

$$Diff_i = F_i - \widehat{F}_i \tag{6}$$

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



(**a**) LGP RSS value(dBm) (**b**) GP RSS value (dBm) (**c**) LDPL RSS value (dBm)

(**d**) LGP RSS uncertainty (dBm) (**e**) GP RSS uncertainty (dBm) (**f**) LDPL RSS uncertainty (dBm)

Figure 4. The estimated RSS values and the uncertainty. LGPL: Log-Distance Path Model.

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 the true radio map less, comparatively.

We also compute the variance over all RPs. The variance is defined in Equation (7).

$$\sigma = \sqrt{\sum_{i=0}^{m} Diff_i^2 / m} \tag{7}$$

We evaluate the average uncertainty for different areas, which are Surveyed Area (SA), Unsurveyed Area (UA) and the Target Area (TA, TA $=$ SA $\cup$ UA). Table 1 illustrates the simulation results.

Table 1. variance for different algorithms in different areas (dBm) SA: Surveyed Area; UA: Unsurveyed Area; TA: Target Area.

Algorithm	SA	UA	TA
GP	1.86	8.09	5.77
LGP	1.88	8.25	5.88
LDPL	6.81	14.53	7.43

From Table 1, we can see that GP has the lowest variance in the target area, which is about 5.77 dBm, followed by LGP with an average of 5.88 dBm. The highest variance comes from LDPL, which is 7.43 dBm. In the surveyed area, all the three algorithms perform better than in the unsurveyed area. In all cases, GP performs the best over the three algorithms.

L is the number of training RPs used for estimating the RSS values for a given virtual node. A large L introduces more training data, and a more accurate result is obtained. However, the time for estimating the RSS values will be increased. In this section, we explore the question of how to find a good balance between the variance and the time for building the virtual database. During the simulation, we set $\lambda = 0.5$, $\varepsilon = 0.5$, and $m = 800$. Figure 5 shows the result.

In this simulation, L increases from 2 to 20. In Figure 5a, we can see that the variance decreases as L increases. Figure 5b shows the time complexity increase with L. If we want to keep a good balance between time complexity and variance, we can set $L = 7$.

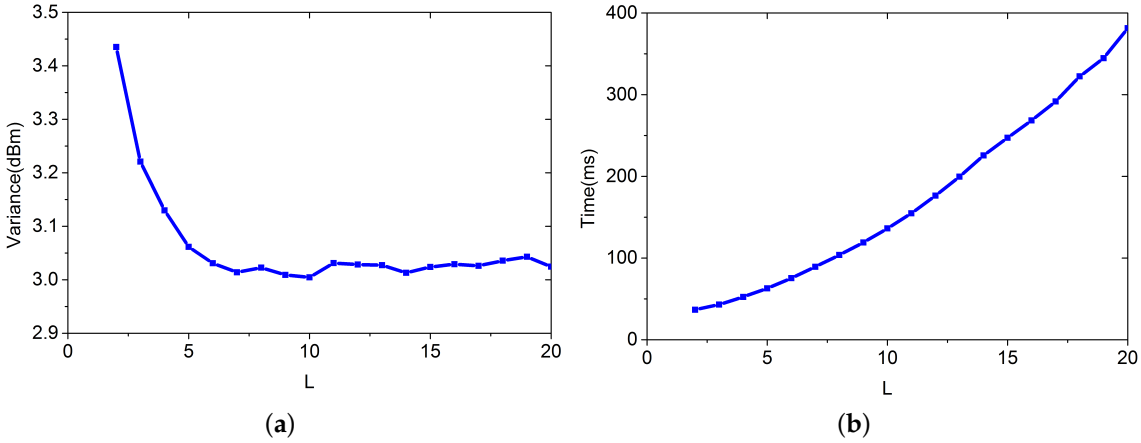


Figure 5. Variance and Time complexity vary with different L. (**a**) Variance of the estimation; (**b**) Time for building the virtual database.

For a more accurate result, we evaluate the three algorithms with different densities of DB^{Crowd}.

In the following simulation, we set ρ^{Crowd} varying from 0.02 to 1, and the training RPs were selected randomly from DB. For each value of ρ^{Crowd}, we simulated 2000 times with $\lambda = 0.5$, $\varepsilon = 0.5$, $m = 800$, and $L = 7$. Figure 6 shows the results. *Time* refers to the time for building the virtual database.

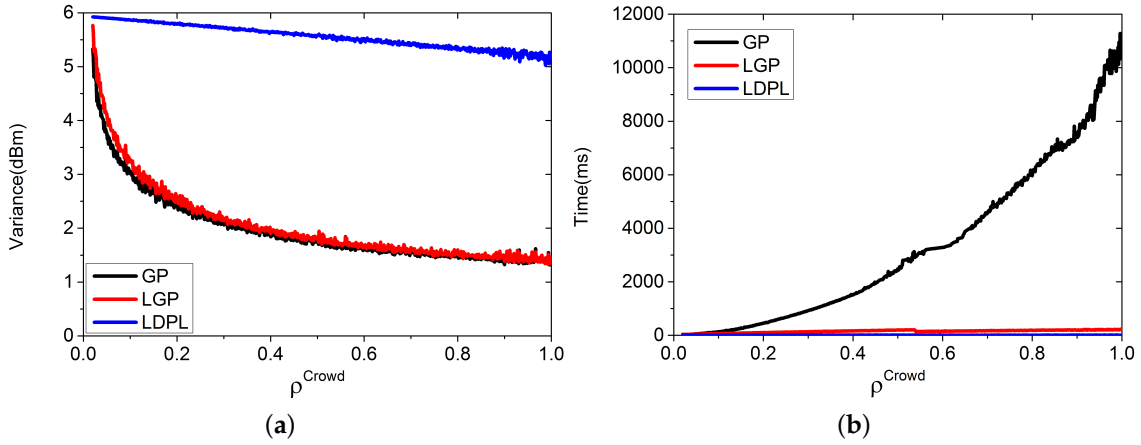


Figure 6. Variance and Time complexity vary with different ρ^{Crowd}. (**a**) Variance of the estimation; (**b**) Time for building the virtual database.

In Figure 6a, GP performs the best, followed by LGP, and LDPL performs the worst. However, the differences between GP and LGP are small. In Figure 6b, LDPL has the lowest time complexity, followed by LGP and GP. In summary, LGP keeps a good balance between variance and time complexity.

3.4. Improved Bayesian Algorithm

Given measurement F_t at time t and $DB^{(v)}$, the objective in fingerprint localization is to estimate the user's real-time coordinate p_t at time t.

The Bayesian localization algorithm is suitable for a user contribution-based localization system for mobile devices [8]. The standard Bayesian localization algorithm will calculate all the RPs' posterior probability, and maximum them to estimate the coordinates. If the fingerprint database

contains a great number of RPs, computing all the RPs' posterior probability would be time consuming. In this paper, we first select K nearest RPs from the virtual database based on the metric defined in Equation (8).

$$d_{t,i} = \sum_{s=1}^{a} (|RSS_{t,s} - RSS_{i,s}|^q)^{1/q} \tag{8}$$

These K RPs have the smallest $d_{t,s}$ among the others in $DB^{(v)}$. A standard Bayesian localization algorithm was used to estimate the user's real-time coordinates based on the selected RPs. The posterior probability of being in one of the selected RPs' locations is given by Equation (9):

$$P(p_i|F_t) = \frac{P(p_i) * P(F_t|p_i)}{\sum_{j=1}^{m} P(p_j) * P(F_t|p_j)} \tag{9}$$

where $P()$ represents the probability density function, $P(p_i)$ is the prior probability of the user's location, and $P(F_t|p_i)$ is the likelihood of observing a set of signal strength measurements F_t at location p_i. p_i is assumed to be uniformly distributed.

For simplicity, we assume that each signal strength $RSS_{i,j1}$ is conditionally independent of every other $RSS_{i,j2}$ for $j1 \neq j2$. So, we have:

$$P(F_t|p_i) = \prod_{j=1}^{a} P(RSS_{t,j}|p_i), i = 1, 2, \ldots, K \tag{10}$$

For modeling the conditional probability $P(RSS_{t,j}|p_i)$, we first show the measurement results from a specified AP at a stationary location in Figure 7.

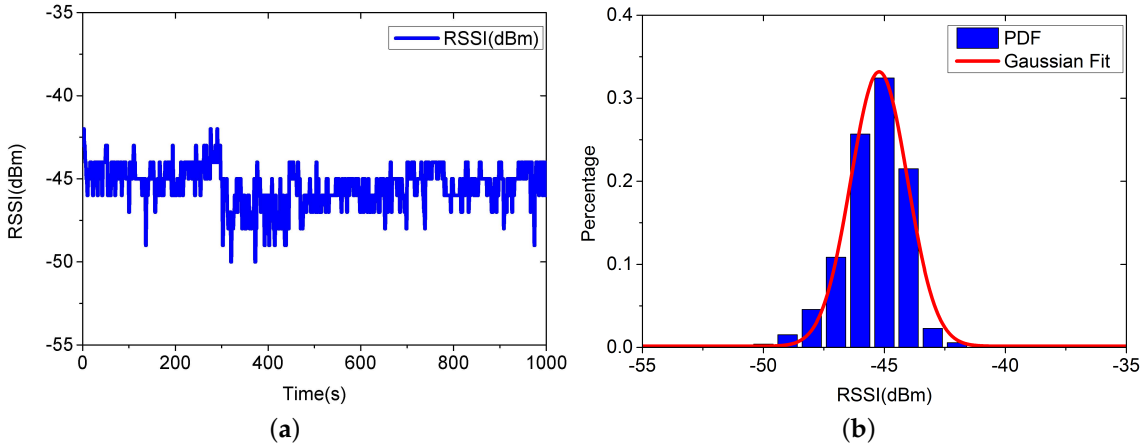


Figure 7. RSSI data and Gaussian Fit. (**a**) RSSI values measured at a stationary location; (**b**) PDF and Gaussian Fit.

Figure 7 implies that we can model the conditional probability $P(RSS_{t,j}|p_i)$ as a Gaussian distribution:

$$P(RSS_{t,j}|p_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(RSS_{t,j} - RSS_{i,j})^2}{2\sigma^2}} \tag{11}$$

The users have to estimate the RSS variance σ_n^2 before uploading the measurement. For the virtual RPs, the variance is given by Equation (5).

Finally, we estimate the user's coordinates using Equation (12):

$$p_t = \sum_{i=1}^{k} p_i \hat{P}(F_t|p_i) \tag{12}$$

where $\hat{P}(F_t|p_i)$ is the normalized conditional probability given by Equation (13):

$$\hat{P}(F_t|p_i) = \frac{P(F_t|p_i)}{\sum_{j=1}^{k} P(F_t|p_j)} \tag{13}$$

4. Results and Discussion

There are some parameters in the proposed algorithm, including λ in MID, ε in LGP, m the number of RPs in virtual database, and K in the improved Bayesian. All of these parameters determine the complexity and positioning accuracy of the proposed algorithm. We first optimize these parameters by simulations, and then we evaluate the proposed algorithm by a real-case scenario experiment.

In the following section, root mean square error (RMSE) is defined in Equation (14):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{W} [(x_i - \widehat{x}_i) + (x_i - \widehat{x}_i)]}{W}} \tag{14}$$

where (x_i, y_i) and $(\widehat{x}_i, \widehat{y}_i)$ are the true and estimated coordinates of the user and W is the number of positioning cases.

4.1. Optimize the Parameters in the Algorithm

4.1.1. λ

The distance λ determines not only the spatial uniformity of the resulting RPs, but also the complexity of the algorithm: by reducing λ, the RPs will be placed more uniformly over the area, but the complexity of MID increases as the number of virtual RPs to be searched increases in inverse proportion to the quad-rate of λ.

In this simulation, we build different $DB^{(v)}$ based on different DB^{Sample} for positioning. The training database are randomly selected from DB, containing 80 RPs. We apply the LGP for estimating the virtual RPs' RSS values. The improved Bayesian algorithm is applied for positioning. The other parameters are set as follows: $\varepsilon = 0.5$, $L = 7$, $m = 80$, and $K = 3$; λ is set to increase from 0.1 to 3.3. Results from 3000 positioning cases are shown in Figure 8.

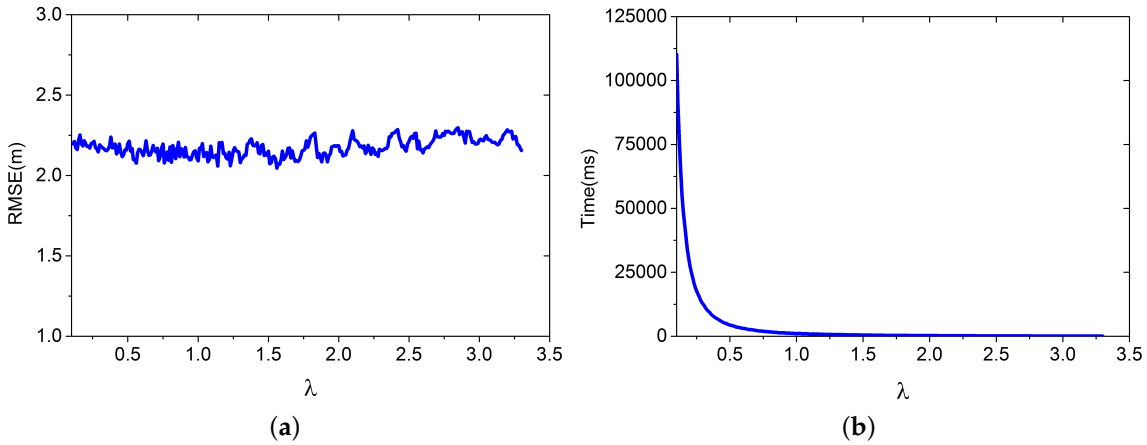


Figure 8. Root mean square error (RMSE) and time for building the virtual database vary with different λ. (**a**) RMSE.; (**b**) time for building the virtual database.

Figure 8a shows that RMSE doesn't change significantly with λ. Figure 8b shows that the time decreases when λ increases. The results from Figure 8 tell us that we can use as large a λ as possible to reduce the time for building the virtual database.

4.1.2. ε

The radius ε has an influence on the positioning accuracy. When the radius is small, the resulting database $DB^{(v)}$ 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 selected radius is large, the resulting database $DB^{(v)}$ will be less spatially uniform, but more training RPs will be present in $DB^{(v)}$.

Similar with the previous setting, we apply the LGP for estimating the virtual RPs' RSS values, and the improved Bayesian algorithm for positioning. The other parameters are set as follows: $\lambda = 3.3$, $L = 7$, $m = 80$, and $K = 3$. We build $DB^{(v)}$ based on the crowdsourcing database, which contains 80 RPs and is randomly selected from DB. ε is set to increase from 0 to 2. Results from 3000 positioning cases are shown in Figure 9.

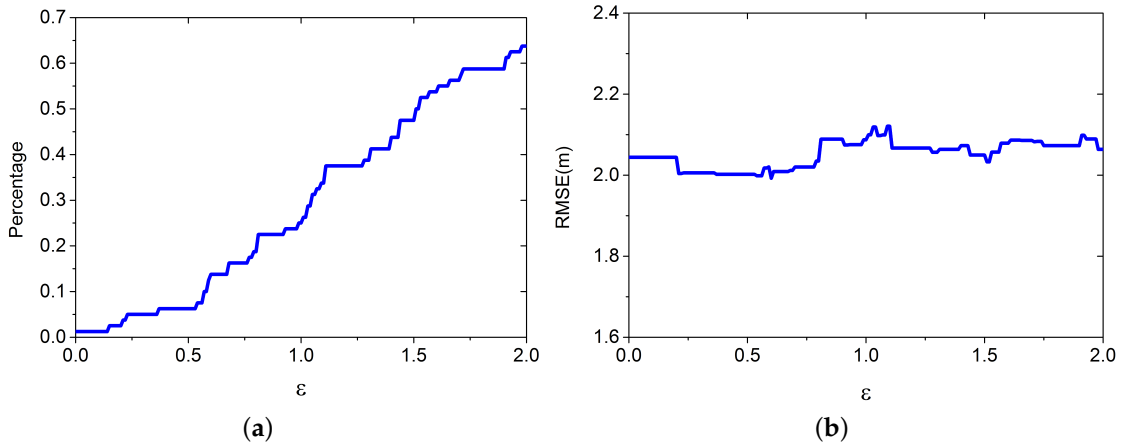


Figure 9. Percentage of training data and RMSE vary with different ε. (**a**) Percentage of training data; (**b**) RMSE.

Figure 9a shows that the training RPs' percentage increase as ε increases. Figure 9b shows that more training data doesn't improve the performance, because $DB^{(v)}$ is less spatially uniform.

4.1.3. 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 improved Bayesian algorithm for positioning. We build $DB^{(v)}$ based on the crowdsourcing database, which contains 80 RPs and is randomly selected from DB. We set K to increase from 1 to 10. The other parameters are set as follows: $\lambda = 3.3$, $L = 7$, $m = 80$, and $\varepsilon = 0.5$. Results from 3000 positioning cases are shown in Figure 10.

The result from Figure 10a shows that using 4 or 5 RPs for positioning performs the best. And Figure 10b shows that the time for positioning is not sensitive to K.

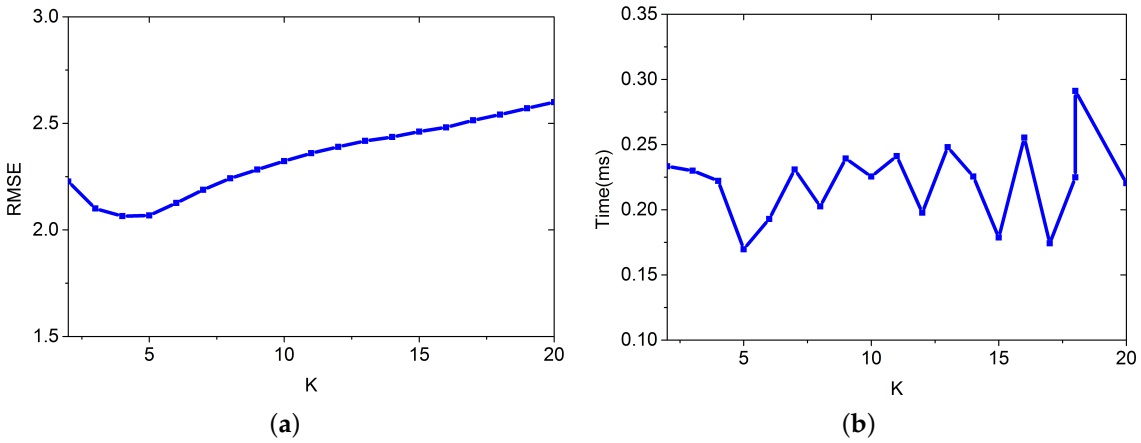


Figure 10. Time for positioning and RMSE vary with K. (**a**) RMSE; (**b**) Time for positioning.

4.1.4. m

m is the number of RPs in the virtual database DB. More RPs might result in a more accurate positioning result, but also needs more time for querying in the database. In this section, we want to find the best size of the virtual database. In this simulation, we set m to increase from 40 to 800, and the training database contains 80 randomly distributed RPs selected from DB. The other parameters are set as follows: $\lambda = 1$, $L = 7$, $K = 5$, and $\varepsilon = 0.5$; For each value of m, the results come from 3000 positioning cases. Figure 11 shows the result.

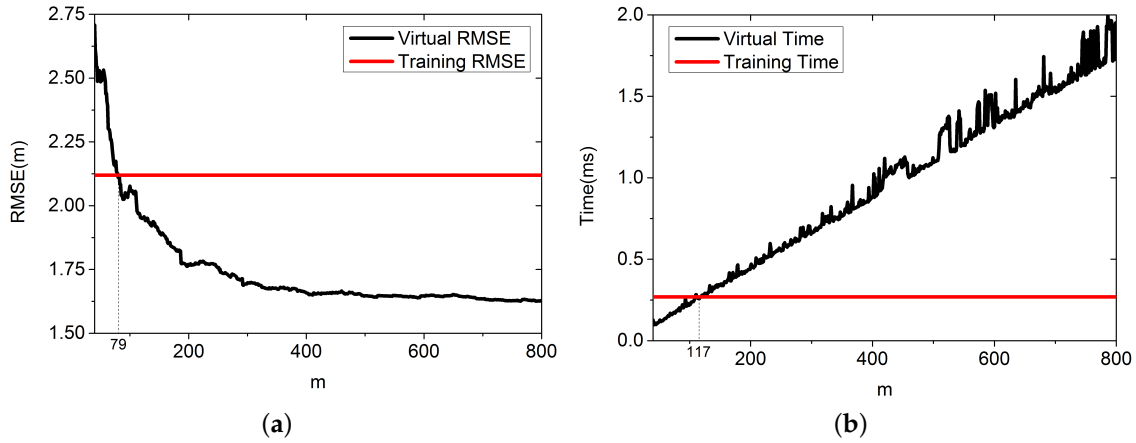


Figure 11. Time for positioning and RMSE with varying m. (**a**) RMSE; (**b**) Time for positioning.

Figure 11a shows that when m is about the same as the number of training RPs, the two methods perform the same. However, as m increases, the proposed algorithm performs better. When $m = 800$, the RMSE is 1.63 m compared with 2.12 m using the training database. The proposed algorithm improves the accuracy by about 23.2%. Figure 11b shows that when $m = 117$, it performs the same as using training database, and the proposed algorithm needs more time for positioning.

4.2. Real-Case Scenario Experiment

For testing the new algorithm in real world, we developed an Android app. The indoor radio map is build in a crowdsourcing way. The user can locate themselves, and they can also upload the fingerprint data to the location server. Figure 12 is the user interface of the app.

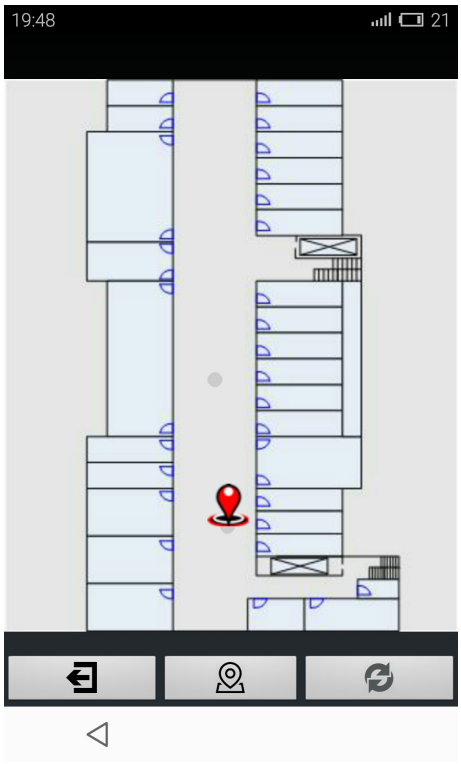


Figure 12. User interface of the app.

In Figure 12, the map is the floor layout of our Lab, covering an area of about 1928 m^2. Clicking the central button will send positioning requirement. Long pressing the interface will change the map of the indoor environment. Double clicking the interface will specify a user's current location. The user can click the right button to send the current RSS measurement and the specified coordinate to the training database.

We first built a training database covering part of the target area. The database contains 71 RPs. We will test the new algorithm in the surveyed area and in different unsurveyed areas. Figure 13 is the distribution of the initial data.

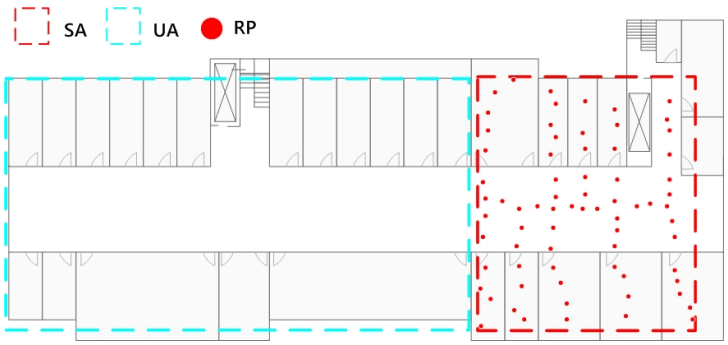


Figure 13. Distribution of initial data.

We first estimate the user's coordinate using virtual database and training database in the surveyed areas. The parameters are set as follows: $\lambda = 1$, $L = 7$, $K = 5$, $\varepsilon = 0.5$. The density of the virtual database is 1. Figure 14 shows the results from 3000 positioning cases.

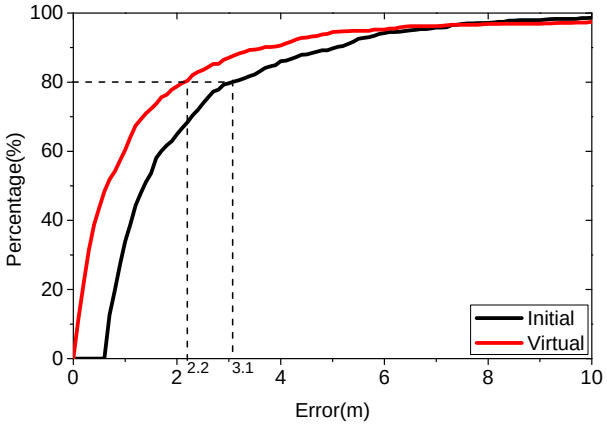


Figure 14. Positioning result using initial database and virtual database in the surveyed area.

Figure 14 shows that the proposed algorithm performs better. The average localization error is 2.47 m using the initial database, while it is 1.84 m using the virtual database. The new algorithm improves the accuracy by 25.5%, with an average positioning error below 2.2 m for 80% of the cases, while the virtual database is 3.1 m.

We make comparison to the standard Bayesian algorithm. Both the new algorithm and the standard algorithm apply the virtual database for positioning. The parameters are set as follows: $\lambda = 1$, $L = 7$, $K = 5$, $\varepsilon = 0.5$. The density of the virtual database is 1. Figure 15 shows the results from 3000 positioning cases.

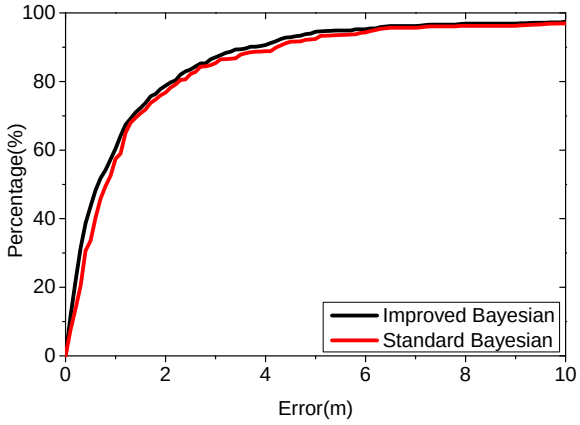


Figure 15. Positioning result using different algorithm.

Figure 15 shows that the new algorithm performs better. The average localization error is 1.84 m using the new algorithm, while the standard is 1.93 m. The new algorithm improves the accuracy by 4.66%, with an average positioning error below 2.2 m for 80% of the cases, while the standard algorithm is 2.3 m.

We evaluate the algorithm in the unsurveyed area. The unsurveyed area is separated into several sub-areas according to the distance to the surveyed area. We compare the positioning accuracy in these sub-areas. Figure 16 shows the experimental results.

Figure 16 illustrates that the RMSE increases as the distance to the surveyed area grows. If the users are less than 10 m away from the surveyed area, the average positioning error is 5.75 m. This positioning result is not accurate enough, but sometimes it is useful, especially for areas without site survey.

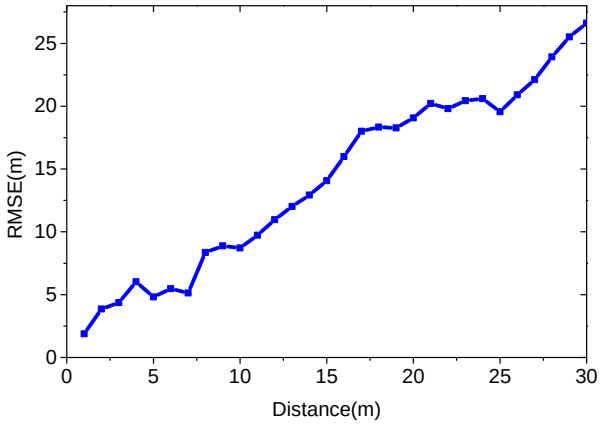


Figure 16. Positioning in different unsurveyed areas using the virtual database.

The proposed algorithm is scalable, which allows the users to continually upload their coordinates to the server to improve the performance of estimation. Figure 17 shows the experimental result. In this experiment, we use the same brand of smartphone for positioning because different devices report network measurement very differently [37].

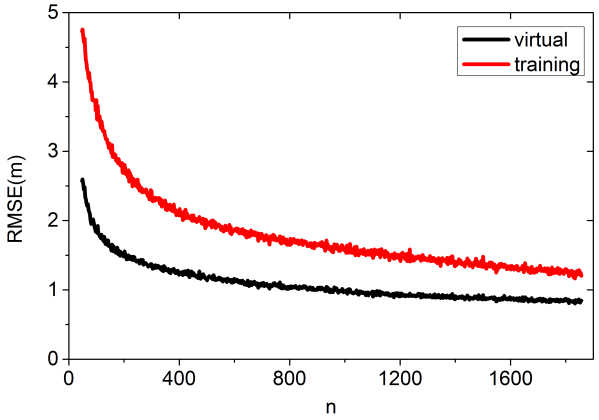


Figure 17. Improving the performance of estimation by crowdsourcing.

In Figure 17, we can see that the proposed algorithm performs better. As the users continually upload their coordinates and RSS measurement, the new algorithm's performance can be improved.

5. Conclusions

The wireless fingerprint technique has the advantages of low deployment cost, supplying reasonable accuracy, and ease of application to mobile devices. As a result, fingerprinting has attracted a lot of attention. In areas without calibration data, however, this algorithm breaks down. Constructing a fingerprint database with high density fingerprint samples is labor-intensive or impossible in some cases. Researchers are continually searching for better algorithms to create and update dense databases more efficiently.

The popularization of smartphones makes mobile crowdsourcing fingerprint localization more practical. However, designing a sustainable incentive mechanism of crowdsourcing remains a challenge. In this paper, we propose a scalable algorithm to create a WLAN radio map by mobile crowdsourcing and Gaussian Process for fingerprint indoor localization. We first propose a Minimum Inverse Distance (MID) algorithm to build a virtual database with uniformly distributed virtual Reference Points (RP). The area covered by the virtual RPs can be larger than the area covered by the

training data. A Local Gaussian Process (LGP) is then applied to estimate the virtual RPs' RSSI values based on the crowdsourced training data. Finally, we improve the Bayesian algorithm to estimate the user's location using the virtual database.

The parameters in the proposed algorithm are optimized by simulations and the new algorithm is tested on real-case scenarios. The average localization error is 2.47 m using the initial database, while the error in the virtual database is 1.84 m. The new algorithm improves the accuracy by 25.5%, with an average positioning error below 2.2 m for 80% of the cases, while the virtual database is 3.1 m. The proposed algorithm also allows the users to continually upload their coordinates to the server to improve the performance of estimation. Moreover, the proposed algorithm can localize the users in the neighboring unsurveyed area. If the users are less than 10 m away from the surveyed area, the average positioning error is 5.75 m.

The proposed algorithm has to rely on a location server. If there is no connection between the server and clients, the user can't upload the positioning requirement. As a result, the client won't receive his coordinate, and this is the problem for all the crowdsourcing fingerprint indoor localization algorithms. Client-based architecture solutions would be more practical. However, with the wide availability of 802.11 WLAN networks, connecting to the internet would be easier. We believe this problem will be solved with the wide deployment of WiFi access points in the future.

Our study requires a strong user collaboration. If the user wants to contribute to the fingerprint database, he should estimate his location with another positioning system and upload the fingerprint, containing the coordinate and the RSS values, to the server. This would lead to the problem that the users are not willing to submit their measurement. For this drawback, we can make the client upload the RSS and location to the server automatically, but the scale of the fingerprint database will increase rapidly. And we are not sure about the reliability of the uploaded data. In that case, we have to filter out the unreliable data. Moreover, we haven't focused on the device diversity problem. It is practically impossible for all the users to have the same brand of smartphone. Our future work will concentrate on these two issues.

Acknowledgments: This study is supported by the NSFC (Natural Science Foundation of China) program under Grand No.71031007, No.61273198.

Author Contributions: Qun Li and Weiping Wang conceived and designed the experiments; Wei Chen performed the experiments; Zesen Shi analyzed the data; Qiang Chang wrote the paper.

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

References

1. Gerstweiler, G.; Vonach, E.; Kaufmann, H. HyMoTrack: A Mobile AR Navigation System for Complex Indoor Environments. *Sensors* **2016**, *16*, doi:10.3390/s16010017.
2. Bahl, P.; Padmanabhan, V.N. Radar: An in-building rf-based user location and tracking system. In Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000), Tel Aviv, Israel, 26–30 March 2000; pp. 750–784.
3. Guzman-Quiros, R.; Martinez-Sala, A.; Gomez-Tornero, J.L.; Garcia-Haro, J. Integration of Directional Antennas in an RSS Fingerprinting-Based Indoor Localization System. *Sensors* **2016**, *16*, doi:10.3390/s16010004.
4. Google Maps, Google Maps Indoor. 2015. Available online: http://www.google.cn/maps/about/partners/indoormaps/ (accessed on 23 May 2015).
5. Ferris, B.; Fox, D.; Lawrence, N. Wifi-slam using Gaussian process latent variable models. *IJCAI* **2007**, *7*, 2480–2485.
6. Du, Y.; Yang, D.; Xiu, C. A Novel Method for Constructing a WIFI Positioning System with Efficient Manpower. *Sensors* **2015**, *15*, 8358–8381.
7. Bolliger, P. Redpin-adaptive, zero-configuration indoor localization through user collaboration. In Proceedings of the First ACM International Workshop on Mobile Entity Localization and Tracking in GPS-Less Environments, New York, NY, USA, 19 September 2008; pp. 55–60.

8. Park, J.-G.; Charrow, B.; Curtis, D.; Battat, J.; Minkov, E.; Hicks, J.; Teller, S.; Ledlie, J. Growing an organic indoor location system. In Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, San Francisco, CA, USA, 15–18 June 2010; pp. 271–284.
9. Rai, A.; Chintalapudi, K.K.; Padmanabhan, V.N.; Sen, R. Zee: Zeroeffort crowdsourcing for indoor localization. In Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, Istanbul, Turkey, 22–26 August 2012; pp. 293–304.
10. Yang, S.; Dessai, P.; Verma, M.; Gerla, M. Freeloc: Calibrationfree crowdsourced indoor localization. In Proceedings of IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 2481–2489.
11. Wan, J.; Liu, J.; Shao, Z.; Vasilakos, A.V.; Imran, M.; Zhou, K. Mobile Crowd Sensing for Traffic Prediction in Internet of Vehicles. *Sensors* **2016**, *16*, doi:10.3390/s16010088.
12. Chang, K.; Han, D. Crowdsourcing-based radio map update automation for Wi-Fi positioning systems. In Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information, Dallas, TX, USA, 4–7 November 2014; pp. 24–31.
13. Lee, M.; Yang, H.; Han, D.; Yu, C. Crowdsourced radiomap for room-level place recognition in urban environment. In Proceedings of the 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Mannheim, Germany, 29 March–2 April 2010; pp. 648–653.
14. Wu, C.; Yang, Z.; Liu, Y. Smartphones based crowdsourcing for indoor localization. *IEEE Trans. Mob. Comput.* **2015**, *14*, 444–457.
15. Rappaport, T.S. *Wireless Communications: Principles and Practice*; Prentice Hall PTR: Lebanon, IN, USA, 2007.
16. Kuo, S.-P.; Tseng, Y.-C. A scrambling method for fingerprint positioning based on temporal diversity and spatial dependency. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 678–684.
17. Barsocchi, P.; Lenzi, S.; Chessa, S.; Furfari, F. Automatic virtual calibration of range-based indoor localization systems. *Wirel. Commun. Mob. Comput.* **2012**, *12*, 1546–1557.
18. LaMarca, A.; Hightower, J.; Smith, I.; Consolvo, S. Self-mapping in 802.11 location systems. *UbiComp* **2005**, *3660*, 87–104.
19. Maher, P.S.; Malaney, R.A. A novel fingerprint location method using ray-tracing. In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2009), Honolulu, HI, USA, 30 November 2009–4 December 2009; pp. 1–5.
20. Raspopoulos, M.; Laoudias, C.; Kanaris, L.; Kokkinis, A.; Panayiotou, C.G.; Stavrou, S. 3D ray tracing for device-independent fingerprint-based positioning in wlans. In Proceedings of the 2012 9th Workshop on Positioning Navigation and Communication (WPNC), Dresden, Germany, 15–16 March 2012; pp. 109–113.
21. Gomez, J.; Tayebi, A.; de Adana, F.M.S.; Gutierrez, O. Localization approach based on ray-tracing including the effect of human shadowing. *Prog. Electromagn. Res. Lett.* **2010**, *15*, 1–11.
22. Dissanayake, M.; Newman, P.; Clark, S.; Durrant-Whyte, H.F.; Csorba, M. A solution to the simultaneous localization and map building (slam) problem. *IEEE Trans. Robot. Autom.* **2001**, *17*, 229–241.
23. Choset, H.; Nagatani, K. Topological simultaneous localization and mapping (slam): Toward exact localization without explicit localization. *IEEE Trans. Robot. Autom.* **2001**, *17*, 125–137.
24. Koyuncu, H.; Yang, S.H. Indoor positioning with virtual fingerprint mapping by using linear and exponential taper functions. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), Manchester, UK, 13–16 October 2013; pp. 1052–1057.
25. Keenan, J.; Motley, A. Radio coverage in buildings. *Br. Telecom Technol. J.* **1990**, *8*, 19–24.
26. Pulkkinen, T.; Roos, T.; Myllymaki, P. Semi-supervised learning for wlan positioning. In Proceedings of the Artificial Neural Networks and Machine Learning-ICANN, Espoo, Finland, 14–17 June 2011; pp. 355–362.
27. Varga, G.; Schulcz, R. Indoor radio location algorithm using empirical propagation models and probability distribution heuristics. *Electr. Eng.* **2013**, *55*, 87–96.
28. Li, L.; Shen, G.; Zhao, C.; Moscibroda, T.; Lin, J.-H.; Zhao, F. Experiencing and handling the diversity in data density and environmental locality in an indoor positioning service. In Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, Maui, HI, USA, 7–11 September 2014; pp. 459–470.
29. Rasmussen, C.E. *Gaussian Processes for Machine Learning*; MIT Press: Cambridge, MA, USA, 2006.
30. Ferris, B.; Haehnel, D.; Fox, D. Gaussian processes for signal strength-based location estimation. In Proceedings of the Robotics Science and Systems, Philadelphia, PA, USA, 16–19 August 2006.

31. Schwaighofer, A.; Grigoras, M.; Tresp, V.; Hoffmann, C. Gpps: A gaussian process positioning system for cellular networks. In Proceedings of the Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013.
32. Chiang, K.-W.; Liao, J.-K.; Tsai, G.-J.; Chang, H.-W. The Performance Analysis of the Map-Aided Fuzzy Decision Tree Based on the Pedestrian Dead Reckoning Algorithm in an Indoor Environment. *Sensors* **2016**, *16*, doi:10.3390/s16010034.
33. Liu, Y.; Dashti, M.; Zhang, J. Indoor localization on mobile phone platforms using embedded inertial sensors. In Proceedings of the 10th Workshop on Positioning Navigation and Communication (WPNC), Dresden, Germany, 20–21 March 2013; pp. 1–5.
34. Chai, W.; Chen, C.; Edwan, E.; Zhang, J.; Loffeld, O. 2D/3D indoor navigation based on multi-sensor assisted pedestrian navigation in wi-fi environments. In Proceedings of the Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS), Helsinki, Finland, 3–4 October 2012; pp. 1–7.
35. AWE. Available online: http://www.awe-communications.com/ (accessed on 16 March 2015).
36. Chintalapudi, K.; Iyer, A.P.; Padmanabhan, V.N. Indoor localization without the pain. In Proceedings of the sixteenth annual international conference on Mobile computing and networking, Chicago, IL, USA, 20–24 September 2010; pp. 173–184.
37. Laoudias, C.; Zeinalipour-Yazti, D.; Panayiotou, C.G. Crowdsourced indoor localization for diverse devices through radiomap fusion. In Proceedings of the 2013 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Montbeliard-Belfort, France, 28–31 October 2013; pp. 1–7.