Crowdsourcing-based WiFi Fingerprint Update for Indoor Localization

Ningjia Fu, Jianzhong Zhang, Wenping Yu, Changhai Wang
College of Computer and Control Engineering, Nankai University
Tianjin, China
funingjia@mail.nankai.edu.cn, zhangjz@nankai.edu.cn
yuwenping@mail.nankai.edu.cn, storm_xp2008@mail.nankai.edu.cn

ABSTRACT

Researches on indoor localization become more and more popular because human spend more life time indoors than outdoors. Among all of the present indoor localization technologies, WiFi fingerprint localization is the most widely used. The method of fingerprint is based on matching the current received signal strength with fingerprints stored in the database to get user's position. This method can get high precision with the simple operation, but it's a labor-intensive work to acquire the fingerprint database which costs much time and human resources. In this paper, we proposed a crowdsourcing method to build an auto-update fingerprint database using the data fed back by numerous users. First we detect the user's step sequence using inertial sensors built in smartphones. Then we establish a Hidden Markov Model (HMM) and propose a Ratio-based Map Matching(RMM) algorithm to match the step sequence with the real path in the map. After the successful match, we bind each fingerprint collected during a walk to its corresponding position, so the auto-update fingerprint database is generated. We did some experiments in a teaching building to evaluate our proposed method, and the results show the accuracy achieved by the method is related to the length of the step sequence. If the step sequence is long enough, the database we generated is very close to the manual measuring results.

Keywords

WiFi fingerprint; crowdsourcing; step sequence; HMM; RMM; indoor localization

1. INTRODUCTION

Position information becomes indispensable in people's life gradually. Global positioning system (GPS) [14] is applied widely in the world, and its satellite coverage reached 98%, enough to acquire people's position information outdoors. But GPS signal strength decreases a lot when it goes through a building due to the multipath fading. So new technologies need to be developed for indoor localization. The study of indoor localization developed rapidly in recent years, such as WiFi, Bluetooth, WLAN, RFID, ZigBee, and Ultrawide Band. WiFi localization is used mostly among all of them. It's for two reasons: one is lots of APs have been installed in shopping malls, office buildings and other large buildings, realizing a complete coverage of WiFi signals. Second, smartphones have built-in WiFi module to receive WiFi signals more easily. Fingerprint localization is the most commonly used in WiFi localization since it has the characteristics of strong stability and high precision. But it is difficult to establish and maintain a WiFi fingerprint database. The fingerprints need professional measurements point by point manually which is time-consuming. With the pass of time, APs will be added/removed in localization environments, namely the WiFi localization environments time-varying, so fingerprints also need a long-term maintenance process.

To avoid the manual maintenance of the fingerprints and update the fingerprint database automatically, the method we proposed in this paper includes four steps: firstly, collect data using inertial sensors in smartphones and generate step sequences; secondly, build our Hidden Markov Model; thirdly, use our Ratio-based Map Matching (RMM) algorithm to match the step sequence with the real path in the map; finally, bind each fingerprint with its position, and store them in the dynamic fingerprint database.

The rest of this paper is organized as follows: Section 2 reviews the related work on crowdsourcing-based WiFi localization. Section 3 introduces the proposed method. Section 4 discusses the evaluation results and analyses of some experiments. The conclusion is given in Section 5.

2. RELATED WORK

There are many methods proposed for achieving high accurate localization in indoor environments. They are mainly divided into two groups: fingerprint-based [1,2,6,8,10–12,16,17,20,21,23–25] and model-based [5, 7, 22]. The fingerprint-based method is used more widely and this paper is also aimed at setting up a fingerprint database.

2.1 Solutions of changeful WiFi signals

The advantage of WiFi fingerprint localization is the popularization of WiFi devices in buildings and smartphones, and the disadvantage is that APs always change after a period of time, which needs to be calibrated and cost much time and human resources. So there are lots of researches to solve this problem.In [17], it uses magnetic field to incorporate WiFi signals to achieve better positioning accuracy. [20] proposes a method called AcMu, an automatic and continuous radio map selfupdating service for wireless indoor localization that exploits the static behaviors of mobile devices.Some systems, such as LANDMARC [15], utilize RFID for indoor localization.

A common way is to install other kinds of wireless devices addi-

ACM TUR-C '17, May 12-14, 2017, Shanghai, China

DOI: http://dx.doi.org/10.1145/3063955.3063989

tionally in big buildings to realize localization [1,10,12,24], such as Bluetooth module [12], ZigBee module [1], etc. These methods can reduce equipment damage speed and the recalibration frequency, but need too much cost if we want a widespread deployment.

Another way is to place other sensors in location environments, for timely delivering the change of the wireless signals in the environments [1] [6] [23], but this kind of method need widespread deployments of new equipments which are costly. [19] uses unexploited RF signal characteristics and leverage users' motions to construct radio floor plan that is previously obtained by site survey.

Some studies [2, 8, 11, 16] are to use the data returned by numerous users to update fingerprints, so that fingerprints can adapt to changeful WiFi environments. This kind of method is called "crowdsourcing". Aiming at the issue of time and manpower cost of fingerprints update, crowdsourcing system allows numerous users to evaluate and correct the positioning result. It makes users not only enjoy the positioning result but also participate in the fingerprint update work. Of course, this method requires users' high participation. Users need to explicitly upload the location and the corresponding fingerprint, and we can't expect users' feedback data are all correct.

2.2 Map Matching Algorithm

There are many methods to match the sensors' data with the map. In the previous work, the Nearest Object Matching (NOM) method is used for matching in [9], which matches the current estimated location to the nearest object. A graph matching algorithm [13] is proposed to build an automatic RSS mapping in complex environments. Some articles construct a multidimensional WiFi fingerprint to improve the accuracy of localization [4] [18]. [3] proposed a crowdsourcing system that users needn't upload data separately. Instead, users only need to open the relevant devices, and the system will automatically match the users' trajectory with the real path by combining PDR and WiFi after users taking a walk. This method can get the matching result even without the initial location. But the premise of this method is that the initial fingerprint database must be measured manually. Distinct from [3], in this paper, from establishment to maintenance of fingerprints are all performed by users. [25] proposed a map matching algorithm based on HMM, utilizing WiFi to calibrate the PDR positioning error. But there will always be cumulative error if you calculate absolute distance by PDR, and the cumulative error increases rapidly with the increase of steps, which leads to incorrect match results. Our RMM algorithm uses the ratio of different paths instead of the actual length in [25].

3. PROPOSED METHOD

Our crowdsourcing system takes the method in [25] for reference, and proposes a new map matching algorithm. As the wide use of smartphones, we still use sensors built in smartphones to collect data. The Hidden Markov Model is chosen because it's suitable for map macthing. Our new map matching algorithm called RMM is to use the ratio relationship instead of the actual length [25] of paths. The reason we use the ratio is that ratios of same attribute datas can offset many system errors and avoid cumulative error by replacing the absolute length with the relative length. The advantage of RMM is that the matching result is obtained by replacing the absolute distance with the step ratio, the error of step length estimation by PDR is removed, and the error caused by the filter or device can be offset by the ratio.

Our system is divided into four parts: step sequences, Hidden Markov Model, RMM algorithm and fingerprint database update. Step sequences are obtained through action recognition algorithm using data collected by inertial sensors in smartphones. HMM is to transform an abstract map to a digital form. RMM algorithm is designed to match the step sequence obtained before with the real map through finding the matching results with the highest probability. After the successful match, the WiFi fingerprints and their coordinates are stored in the fingerprint database, to complete the final part called fingerprint database update.

Preparation work: pretreatment of a physical map by storing relevant information in the form of array, a smartphone with the built-in accelerometer, gyroscope, magnetometer and WiFi. Our crowdsourcing system greatly reduces the user's operation difficulty when they upload data, because they needn't measure and upload point by point. RMM algorithm can still match the best path even without knowing the initial point. It also reduces workload of staffs because the fingerprint can be established and maintained by users completely. From the above, our method solves many problems that other crowdsourcing system can't avoid.

3.1 Step Sequence

The step sequence means a series of continuous and orderly turnings that include two parts: direction and step. There is a straight path between two adjacent corners, including its walking direction and step, shown in Fig.1. A series of straight paths constitute our step sequences. In this paper, with the help of accelerometer, gyroscope and magnetometer in the smartphone, we complete the collection work of direction and step.

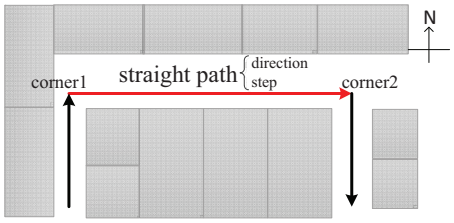


Figure 1 Example of straight path.

Accelerometer is used for step counting. People's steps have periodicity during them walking as well as the acceleration data collected at the same time. Accelerometer returns three values, representing the acceleration on three axes of its own coordinate system. Here we take vector sum of the three as the effective acceleration. By this value we can get two messages: the action (walk or stand), the steps if walking. The method to distinguish walking and standing is to set a threshold acceleration, this paper we set it to 10.5 based on the experiments. We judge the action as walking if the acceleration exceeds threshold. The step is detected by peak detection algorithm proposed in [25] and filtered through threshold method. Each peak represents a step, so the total number of peaks means the walking steps. The step detection result is shown in Fig.2 (a).

We assume that pedestrians only take *normal turn* and *U_turn*. *Normal turn* means turn a right angle, and *U_turn* means turn around. This assumption is reasonable because it's based on the fact that buildings are generally designed for the rightangle corner. Every corner in the map is defined as a node. Whether the pedestrian turns or not is detected by gyroscope. Gyroscope returns the angular velocity of the three axes. The returned value is positive for left turn, negative for right turn, zero for straight. The heading is measured by accelerometer and magnetometer. Particularly, this heading is the absolute value that presents the angle between the device coordinate system and the earth coordinate system. The

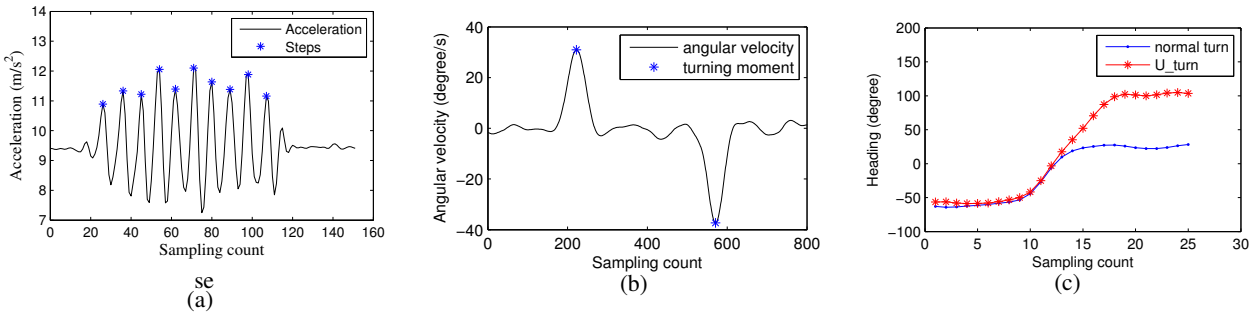


Figure 2 Step detection, angular velocity detection, and heading change.

change of heading is used to judge *normal turn* or *U_turn*. The feedback data of the three sensors will change obviously when pedestrians turn, which is shown in Fig.2 (b) and (c).

When the pedestrian walks along a straight path, the angular velocity of the central axis is 0. Then the angular velocity increases first and then decreases during a left turn, shown as a peak in Fig.2 (b), while the right turn is shown as a nadir. By the number and order of peaks and nadirs we can get the turn sequence.

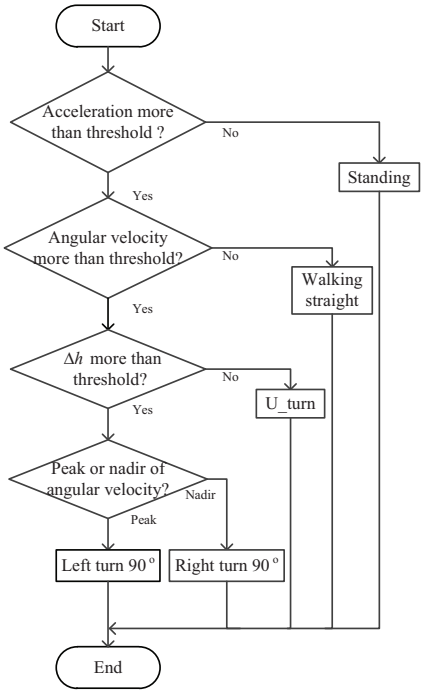


Figure 3 Action recognition algorithm.

When a turn is detected, we need to distinguish *normal turn* and *U_turn*. The difference of them is shown in Fig.2 (c). Here we use the gyroscope and magnetometer commonly to get the heading data. Referencing the AD detection method proposed in [25], we use the following formula to distinguish them:

$$\begin{cases} normal\ turn, if\ \Delta h < H_t \\ U\_turn, \quad otherwise \end{cases} \tag{1}$$

Where $\Delta h = abs(mean(h(T - t_{win} : T)) - mean(h(T : T + t_{win})))$, h means the heading measured by smartphones, H_t means the threshold to judge *normal turn* or *U_turn*. t_{win} means the time window for calculating the average, and T means the turning moment. Referencing to [25], here we set t_{win} to 1.5s and H_t to 135°.

Our direction detection result is recorded as $\{1, 2, 3, 4\}$, which represent the direction of east, south, west and north. The initial direction is known through the heading data. So the next direction can be determined by the initial direction and the step sequence that we get through action recognition algorithm.The action recognition algorithm is shown in Fig.3.

In order to increase the accuracy of direction and step detection, we use a 4-order Butterworth low-pass filter with the 10Hz cutoff frequency to preprocess the feedback data. The filter can decrease the influence of the noise caused by the jitter of the human body.

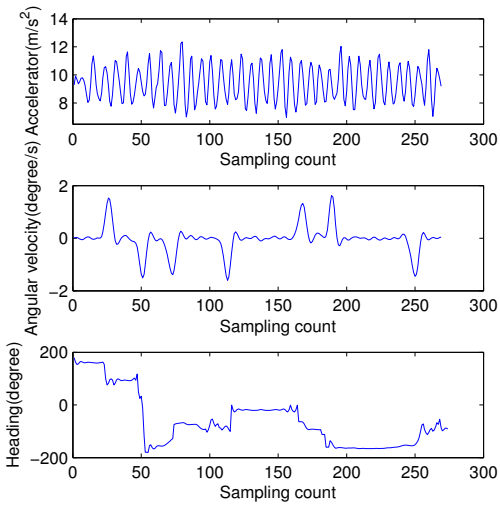


Figure 4 Example of a step sequence.

After the step and direction detection, we get the step sequence as follows.

$$S\_Sq = \{d_0, s_0; d_1, s_1; d_2, s_2; \ldots\ldots\}$$

Where d_i means the direction of the ith straight path, and s_i means the step counting result on the direction of d_i. Pairs of (d_i, s_i) constitute the step sequence. We delete the first and the last straight path in S_Sq because they are not complete. And if an *U_turn* is detected, it means pedestrians turn around to the opposite direction,

and we need to merge the adjacent data of the U_turn by subtracting the steps on two opposite directions. An example of S_Sq is shown in Fig.4.

In this paper, the step is used instead of the distance between two nodes. In section 3.4 we introduced a Ratio-based Map Matching algorithm that using the ratio of different paths' length in order to reduce the accumulative error caused by absolute distance by PDR, so here we use the step ratio directly rather than calculate the absolute distance. In other words, our method doesn't need to use actual walking distance, only needs to know the length ratio of each straight path.

3.2 Indoor Map Processing

Based on the detected step sequence, the pedestrian's trajectory can be determined by matching these turns to the corresponding nodes of the real map.

Before building our mathematical model, we need to process a physical map firstly. Buildings have their own topologies. Walls, tables and chairs limit the direction and distance when you walk in a building. The trajectory of the pedestrian can't violate the structure of the building. We define the corner as "node" where people can make a turn. Each node has some properties, including:

1) Coordinate
2) Feasible direction
3) Adjacent nodes of each feasible direction

There are only four candidates of feasible directions including east, south, west and north. Pedestrians can only transfer between the nodes on the same straight path. Nodes that can be reached along a straight path are known as "adjacent nodes". Fig.5 (a) shows an example of a node and its properties. Node 1, 3 and 4 are all adjacent nodes of node 2.

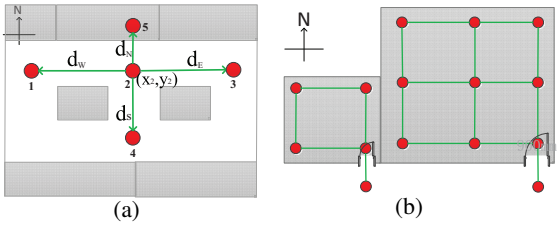


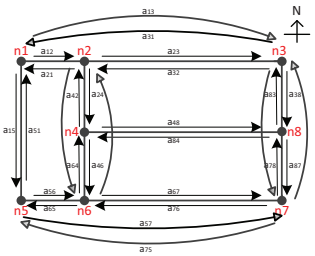
Figure 5 Properties and distribution of nodes.

We design 4 nodes for the small room that less than $50m^2$, like office rooms, and 9 nodes for the big room that more than $50m^2$, like meeting rooms. The distribution of the node is shown in Fig.5 (b).

3.3 Hidden Markov Model

HMM is a statistical model. The most obvious character of HMM is the hidden state in the corresponding Markov process. The difficulty is how to find the hidden state sequence with the maximum probability through the observable state. The transition probability between the hidden states forms the transition probability matrix. The observable state probability of a specific hidden state forms the confusion matrix. The structure of HMM is similar to that of step sequences detecting,

In this section, we take the indoor road network (seen in Fig.6) as an example to establish the HMM, which is used to match the physical map and the step sequence.



	n1	n2	n3	n4	n5	n6	n7	n8
n1	0	1/3	1/3	0	1/3	0	0	0
n2	1/4	0	1/4	1/4	0	1/4	0	0
n3	1/4	1/4	0	0	0	0	1/4	1/4
n4	0	1/3	0	0	0	1/3	0	1/3
n5	1/3	0	0	0	0	1/3	1/3	0
n6	0	1/4	0	1/4	1/4	0	1/4	0
n7	0	0	1/4	0	1/4	1/4	0	1/4
n8	0	0	1/3	1/3	0	0	1/3	0

Figure 6 Example of indoor road network and corresponding transition probabilities.

We use the array $\{H,\ O,\ \pi,\ A,\ B\}$ to describe a Hidden Markov Model. $H = (h_1,\ h_2,\ h_3,\ \cdots, h_N)$ is a set of hidden states, and N is the number of hidden states. $O = (o_1,\ o_2,\ o_3,\ \cdots\ o_M)$ is a set of observable states, and M is the number of observable states. $\pi = \{\pi_i\}$ represents the initial state distribution. $A = \{a_{ij}\}$ represents the transition probability matrix, $a_{ij} = P(x_t = h_j | x_{t-1} = h_i)$. $B = \{b_i(k)\}$ represents the confusion matrix corresponding to the kth hidden state, $b_i(k) = P(o_t(k)|x_t = h_i)$.

In this paper, we can't directly know which node pedestrians walking by only from the data of the accelerometer, gyroscope, and magnetometer. What we get is only the step sequence. So the observation state in our HMM is the step sequence, and the hidden state is the node in the road network. We build the HMM as follows:

1) Hidden states: Nodes in road network. The moving from a node to its adjacent node is the transition between two hidden states.

2) Observable states: The step sequence we get from Section 3.1 contains direction and step information. One step with its corresponding direction constitutes an observable state, so that the step sequence is divided into a chain of observable state one by one, as follows:

$$S\_Sq = \{(d_0, s_0); (d_1, s_1); (d_2, s_2); \ldots\ldots\}$$

We need to process the step data in S_Sq to change them into the ratio form. The method is to replace s_i with $sr_i = s_i/s_0,\ i = 0, 1, 2\cdots$. So the S_Sq becomes:

$$S\_Sq = \{(d_0, sr_0); (d_1, sr_1); (d_2, sr_2); \ldots\ldots\}$$

3) Transition probability matrix: At the beginning, the transition probability between the adjacent nodes is set to be uniform, and the rest is 0. An example is shown in Fig.6. Transition probability matrix will be updated regularly. According to the uploading frequency of each path, replace the current transition probability with the uploading frequency that updated by crowdsourcing.

4) Confusion matrix: The confusion matrix records the probability distribution of one or more observable states of each hidden state. In our HMM, there are two kinds of observation states: step ratio (sr) and direction. The element in confusion matrix is called emission probability, defined as:

$$P(sr_t, d_t|h_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(sr_t - \frac{l_i}{l_0})} \tag{2}$$

$P(sr_t, d_t|h_i)$ means the probability distribution of observable state sr_t on the direction of d_t of the current hidden state h_i. This probability is assumed to be Gauss distribution so it goes as the formula (2), σ for the measuring standard deviation of step counting. l_i is

the distance between the current detected node and the last one, calculating by coordinates of two nodes. l_0 means the length of the first detected path. The closer the sr and l/l_0, the larger the P.

5) Initial state distribution: We don't know the initial location of the pedestrian in the paper, so we assume that the step sequence takes all nodes as the initial location with the equal probability. It is the initial probability distribution we use.

Algorithm 1: RMM

input: Indoor road network: IRN;
A, B, π, N of HMM;
Step sequence: S_Sq.
output: The trace with the maximum probability.
definition:(global) $trace\_array$ = all the traces detected in process;
P_e = the emission probability;
P_t = the probability of each trace in $trace\_array$;
$neighbor_{i,d_j}$ = adjacent nodes of n_i on the j th direction.

```
for i = 1 : N
    trace(1) = i; j = 1;
    search(i, j, trace);
end for
calculate P_t;
trace_m = arg max (P_t)   (trace ∈ trace_array)
output trace_m.
function search(i, j, trace)
    if j > length(S_Sq);
        add trace to trace_array;
        return;
    else
        get neighbor_{i,d_j};
        number=length(neighbor_{i,d_j});
        if number == 0
            delete trace;
            return;
        else
            j = j + 1; k = 0;
            while k ≠ number
                k = k + 1; trace(j) = neighbor_{i,d_j}(k);
                calculate P_e;
                if P_e > 0.7
                    search(neighbor_{i,d_j}(k), j, trace);
                else
                    delete trace(j);
                end if
            end while
        end if
    end if
end function
```

3.4 Ratio-based Map Matching Algorithm

How to find the hidden state sequence with the maximum probability?

We proposed a Ratio-based Map Matching (RMM) algorithm with the input of step sequences to do this work. RMM algorithm needs to search every node to find the initial hidden state. The search criterion is the observable state in S_Sq. Taking the map in Fig.6 as an example, if the first observable direction is east, the nodes of n1, n2, n4, n5, n6 will be selected out, as the first layer of the path tree. Here we take n1 as the root to build a path tree seen in Fig.7. Then search the next layer relying on the next observable direction. Definitely, nodes on the current layer must be adjacent to the node on last layer. The path tree is established completely until the last observable state is used. After searching out a node, we calculate the emission probability by the formula (2). Only when the emission probability is bigger than the threshold, this path can continue to search for the next layer. In this paper, we set the threshold to 0.7 based on the experiments. There are two conditions to continue a path: the existence of adjacent node, the large enough emission probability. These two limits greatly reduce the branches of the path tree and reduce the time complexity.

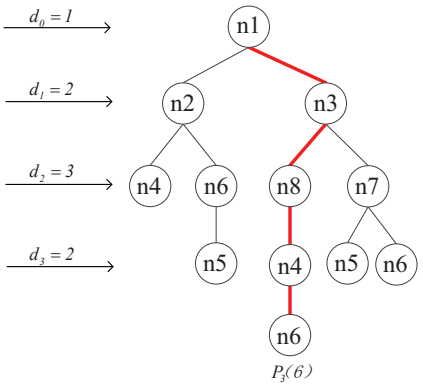


Figure 7 Example of path tree.

According to the complex indoor road network, it's nearly impossible that two paths have the same shape. That is to say, there is only one longest branch of the path tree. But it is only applicable to the situation that the step sequence is long enough. If the step sequence is very short, the information can be used to match is not enough, so it may gain two or more longest branches. Considering of this situation, we select the maximum probability as the output. The probability is calculated as follows, where π, a, b are all mentioned in Section 3.3.

$$P_0(j) = \pi(j) b_j(k_0) \qquad 1 \le j \le N \tag{3}$$

$$P_t(j) = \sum_{t=1}^{n} (P_{t-1}(i) a_{ij}) b_j(k_t) \qquad 1 \le i, j \le N \tag{4}$$

Our method is summarized with the pseudocode in Algorithm 1.

Algorithm 1 is to search the hidden nodes one by one through a recursive method. The nodes have to satisfied two conditions, the adjacent node and probability, otherwise it will be delete. Our method needn't know the start point, and can't locate the end point. It is used for the construction and update of the dynamic WiFi fingerprint database described in Section 3.5 below.

3.5 Fingerprint Database Update

Smartphones have the function of collecting WiFi RSS currently. In this paper, from start to end of the walk, pedestrians need to collect WiFi signals continuously. A WiFi fingerprint is a set of pairs: the received signal strength (RSS) and MAC address of the AP. A fingerprint database stores fingerprints with their corresponding locations.

The WiFi signal collected by smartphones is associated with the step sequence by timestamps. The turning moment is already known, so there are one starting moment and one ending moment

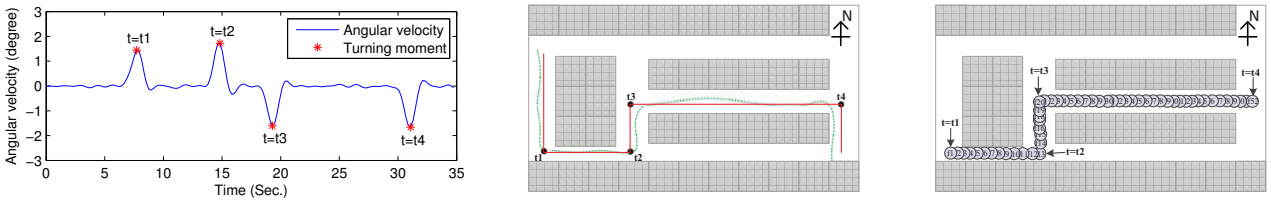


Figure 8 Distribution of nodes in rooms.

of each straight path. Find all the WiFi fingerprints in these two moments, and deploy them on the matched path in equal interval. In this way each WiFi fingerprint has its corresponding coordinate. Then we bind the fingerprint and its location, and store them in the database. The method is shown in Fig.8.

The fingerprint database without the calibration of staffs can update itself relying on the feedback data by pedestrians. This method can ensure accuracy even if a new AP installed or an old AP removed. The database is updated quantificationally. If the number of new uploaded fingerprints reach to a specified number, we calculate an average value by all of them. And then add this value and the origin value with the same weight to get the update value. The weight is set to (1/2, 1/2) as the following formula:

$$value_u = \frac{1}{2}(value_o + \frac{1}{N} \cdot \sum_{i=1}^{N} value_{n_i}) \tag{5}$$

Where $value_u$ means the updated value , $value_o$ means the origin value, $value_n$ means the new uploaded value and N is the total number of $value_n$. By this way, the weight of the old fingerprint which has long survival period has been weakened, and the aim of replacing the old fingerprint by the new fingerprint is achieved.

In addition, if the fingerprint database only increases over time, it will be too large to affect the localization efficiency. To this purpose, the database will be filtered on a regular time interval. The higher the frequency is, the longer the fingerprint survives. If a fingerprint is unused for a long time, it will be deleted in the database.

4. EVALUATION

4.1 Accuracy of Direction and Step Detection

In order to prove the accuracy of the direction and step detection result of the action recognition algorithm, we have done some experiments. The selected experimental site is 50m*50m teaching building. The selected device is a smartphone on version 4.1.1 Android system that has built-in accelerator, gyroscope and magnetometer. The sensor sampling frequency is set to 20 Hz.

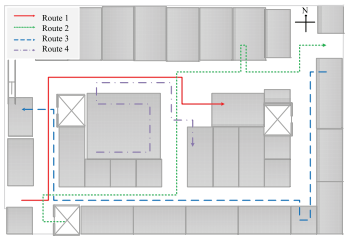


Figure 9 Experiment routes of direction and step detection.

Four participants including two males and two females are asked to walk through the 4 defined routes in Fig.9. Each route is repeated 10 times per participant. When testing, pedestrians hold the smartphones horizontally in front of the chest, and keep walking at a uniform speed. The starting point and end point must be in the road network. We recorded the actual steps on each straight path during them walking and the label of nodes they passed through. Then we calculated the sequence of step and direction using the feedback data of sensors through Algorithm 1. But before run Algorithm 1, we processed the data through the 4-order low-pass Butterworth filter, cutoff frequency 3 Hz, to remove the interference extremum. After running Algorithm 1, we got the step sequence output. It is compared with the actual value we recorded during walking to get the accuracy of step and direction detection. The result is shown in Tab.1.

We calculate the average of the detected turns for each route, and the average of the detected step results for each straight path of each route. For turn detection, we can see that the accuracy of Route 2 and 3 can't reach to 100%, because an *U_turn* exists in these two routes. In our detection results, the *U_turn* is not completely able to be detected well. It is often mistaken for *normal turn*. For Route 2, there are totally 40 sets of data collected which means 40 *U_turn*s should be detected, but only 22 of the groups are detected correctly which leads the accuracy reducing from 100% to 98.2%, the same reason for Route 3. But for *normal turn*, detection result is very accurate, which can reach to 100%.

For step detection, we use the formula (6) to obtain the similarity to evaluate its accuracy.

$$S_{r,d} = \frac{1}{|N|} \sum_{\forall i \in N} \frac{\min(|s\_real(i)|, |s\_det(i)|)}{\max(|s\_real(i)|, |s\_det(i)|)} \tag{6}$$

Where N is the total number of straight paths in each route, *s_real* means the real step of one straight path and *s_det* means the detected step of the same straight path. $S_{r,d}$ means the similarity between the real value and the detected value of one straight path. The range of $S_{r,d}$ is from 0 to 1. And $S = 1$ means exactly the same.

From the result shown in Tab.1, we can see that there is no more than 5 steps error between the detected step and the real step. The minimum similarity is 0.872 and the rests are all above 0.9. The high similarity shows that the step can be detected accurately based on the proposed method using a smartphone.

4.2 Matching Result by RMM

Matching algorithm and the length of the step sequence both affect the accuracy of matching results. Generally, the longer the step sequence, the more accurate the matching result, because it can bring more information to match. In this section, we select 10 long routes to prove the effect of the length of the step sequence on the matching accuracy. They all have 20 turns. For each long route, at the beginning, we use the first straight path to get the matching result, and then add one by one until the route is finished. Four participates still hold the smartphones on their hands and walk through

Table 1 Experiment results of direction and step detection.

	Turn			Step		
Route	Real	Detected	Accuracy	Real	Detected	Similarity
1	4	4	100%	66,44,13	60,30,13	0.962
2	9	9	98.2%	10,67,60,40,1,39,9	11,67,60,42,2,40,9	0.905
3	5	5	98.5%	29,31,8	28,32,9	0.941
4	7	7	100%	20,9,21,3,31,16	20,10,24,5,30,18	0.872

each route twice. The matching result is shown in Fig.10. In this process, the length of the step sequence is increasing, the information that can be used to match is increasing as well.

In order to show the difference between RMM and PDR. We also estimate the step length using the collected data through PDR algorithm. The product of the step length and the step number is the absolute distance (i.e., estimated length) of each straight path, that instead of the ratio as the input of the algorithm. The result is shown in Fig.10.

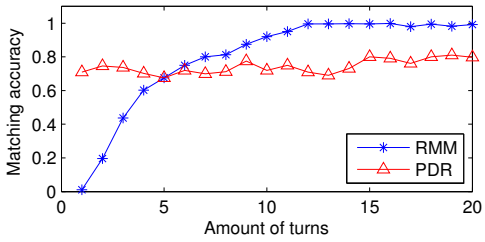


Figure 10 Matching results of RMM and PDR.

The PDR method is to match each straight path in the map directly by the absolute distance, which is independent of the other straight paths. The information that used to match is direction and absolute distance. But our RMM method utilizes the ratio relation between the straight path and its adjacent straight path. The increase in the amount of turns means the number of steps increases. As is concerned in the PDR method, the cumulative error always exists. So the matching result always has a gap (almost 0.3 percent) from the correct path of PDR. It determines that with the increase of steps and turns, the matching error of PDR acts out some small fluctuations due to the cumulative error and useful information increases, otherwise, the matching result of RMM is better and better with the increase of availability information (see in Fig.10).

When the amount is less than 5, the accuracy of PDR is greater than RMM. It is because the useful information for RMM is too little to match a correct path especially when there are only one or two turns. As the amount greater than 5, the accuracy of RMM exceeds PDR. These results show that our method requires the turns that pedestrians walking by, are sufficiently rich.

4.3 Fingerprint Database Performance

According to the fingerprint database update method mentioned in Section 3.5, we did some experiments to verify the relationship between the accuracy of the auto-generated fingerprint and the number of uploaded fingerprint. The accuracy is obtained by calculating the Euclidean distance (ED) of the autoupdated and manual-measured data.

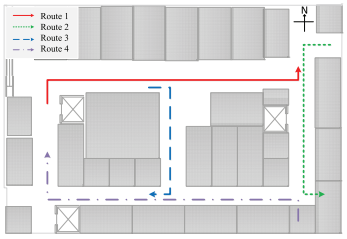


Figure 11 Experiment routes of RMM algorithm.

We selected 4 straight paths (see in Fig.11) to measure the WiFi RSS in an interval of 0.5 meters, and recorded the coordinate of the corresponding measurement location, thus the actual fingerprint was formed. This work was done by every participant walking each path 20 times holding the smartphones on their hands. The WiFi sampling frequency of the smartphone is set to 1Hz. After the procedure of action detection, we deployed WiFi fingerprints to the walking path using timestamps of the action-related data. Thus each fingerprint has a coordinate. A total of 80 WiFi fingerprints were collected of each path, and we uploaded them cumulatively, increase of five every time. We updated the database using the method introduced in Section 3.5 after each upload, and calculate the ED of the updated result and manual-measured result as the fingerprint database performance which is shown in Fig.12.

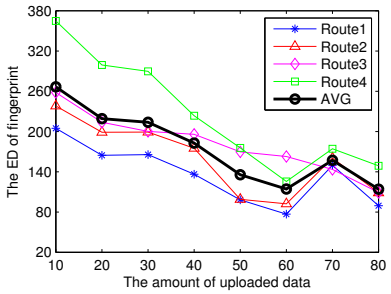


Figure 12 Fingerprint database performance.

After testing, the experiment site has a total of 45 APs. In an interval of 5 sets of collected WiFi RSS to upload, we got the line graph in Fig.12. Each time we uploaded, we got ED of the new updated database and the existing measured data (seen as standard data). We use the ED calculated by formula (7) to express the accuracy.

$$ED = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(RSS_i - RSS\_S_i)^2} \tag{7}$$

Where RSS means the new updated fingerprint and RSS_S means the standard RSS recorded before, N means the totally number of APs which is set to 45 here. ED represents the similar degree between the updated one and the standard one. The smaller ED means the bigger similarity.

We got 5 curves in Fig.12. Four of them represents the ED of 4 routes and the last one represents the average ED of these four. For the first 4 curves, each point in the curve represents a new ED after each update for each route. The last black curve is the average of the previous 4 curves. From Fig.12, we can see the overall trend of ED is reduced with the increase of the amount of uploaded data. At the amount of 60, the ED is the minimum which reaches to 78.91. We can see at the amount of 70, the ED increases suddenly because there are more incorrect data among the new uploaded data which lead to the lower accuracy of fingerprints. The results mean the auto-updated database can be a good substitute for manual measured database when the uploaded data amount is greater than 60.

5. CONCLUSION

In this paper, we proposed a "crowdsourcing" method to generate the auto-updated WiFi fingerprint database. The method is based on the data collected by smartphones and the action recognition algorithm to obtain pedestrians' step sequence. And then find the most likely path by RMM algorithm, thus the mapping relation between the WiFi fingerprint and the geographical position is obtained, and the auto-updated fingerprint database is generated. The method in the conditions of unknowing the starting position can still get high accuracy of the matching results. Through some experiments, it is indicated that the method can achieve the similar performance with manual calibration.

6. REFERENCES

[1] H.-S. Ahn and W. Yu. Environmental-adaptive rssi-based indoor localization. *Automation Science and Engineering, IEEE Transactions on*, 6(4):626–633, 2009.

[2] A. Barry, B. Fisher, and M. L. Chang. A long-duration study of user-trained 802.11 localization. In *Mobile Entity Localization and Tracking in GPS-less Environnments*, pages 197–212. Springer, 2009.

[3] K. Chang and D. Han. 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*, pages 24–31. ACM, 2014.

[4] D. Chen, L. Du, Z. Jiang, W. Xi, J. Han, K. Zhao, J. Zhao, Z. Wang, and R. Li. A fine-grained indoor localization using multidimensional wi-fi fingerprinting. In *Parallel and Distributed Systems (ICPADS), 2014 20th IEEE International Conference on*, pages 494–501. IEEE, 2014.

[5] H.-C. Chen, T.-H. Lin, H. Kung, C.-K. Lin, and Y. Gwon. Determining rf angle of arrival using cots antenna arrays: a field evaluation. In *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*, pages 1–6. IEEE, 2012.

[6] Y.-C. Chen, J.-R. Chiang, H.-h. Chu, P. Huang, and A. W. Tsui. Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics. In *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 118–125. ACM, 2005.

[7] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 173–184. ACM, 2010.

[8] T. Gallagher, B. Li, A. G. Dempster, and C. Rizos. Database updating through user feedback in fingerprint-based wi-fi location systems. In *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2010*, pages 1–8. IEEE, 2010.

[9] D. Gusenbauer, C. Isert, and J. Krösche. Self-contained indoor positioning on off-the-shelf mobile devices. In *Indoor positioning and indoor navigation (IPIN), 2010 international conference on*, pages 1–9. IEEE, 2010.

[10] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose. Mapping and localization with rfid technology. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 1015–1020. IEEE, 2004.

[11] R. Hansen, R. Wind, C. S. Jensen, and B. Thomsen. Algorithmic strategies for adapting 802.11 location fingerprinting to environmental changes. In *International Conference on Indoor Positioning and Indoor Navigation*. Citeseer, 2010.

[12] A. Hossain and W.-S. Soh. A comprehensive study of bluetooth signal parameters for localization. In *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pages 1–5. IEEE, 2007.

[13] Z. Jiang, J. Zhao, J. Han, S. Tang, J. Zhao, and W. Xi. Wi-fi fingerprint based indoor localization without indoor space measurement. In *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*, pages 384–392. IEEE, 2013.

[14] J. Liu, B. Priyantha, T. Hart, H. S. Ramos, A. A. Loureiro, and Q. Wang. Energy efficient gps sensing with cloud offloading. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 85–98. ACM, 2012.

[15] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. Landmarc: indoor location sensing using active rfid. *Wireless networks*, 10(6):701–710, 2004.

[16] J.-g. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie. Growing an organic indoor location system. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 271–284. ACM, 2010.

[17] Y. Shu, C. Bo, G. Shen, C. Zhao, L. Li, and F. Zhao. Magicol: Indoor localization using pervasive magnetic field and opportunistic wifi sensing. *IEEE Journal on Selected Areas in Communications*, 33(7):1443–1457, 2015.

[18] C. Wu, Z. Yang, and Y. Liu. Smartphones based crowdsourcing for indoor localization. *Mobile Computing, IEEE Transactions on*, 14(2):444–457, 2015.

[19] C. Wu, Z. Yang, Y. Liu, and W. Xi. Will: Wireless indoor localization without site survey. *IEEE Transactions on Parallel and Distributed Systems*, 24(4):839–848, 2013.

[20] C. Wu, Z. Yang, C. Xiao, C. Yang, Y. Liu, and M. Liu. Static power of mobile devices: Self-updating radio maps for wireless indoor localization. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2497–2505. IEEE, 2015.

[21] J. Xiong and K. Jamieson. Arraytrack: a fine-grained indoor location system. In *Presented as part of the 10th USENIX*

Symposium on Networked Systems Design and Implementation (NSDI 13), pages 71–84, 2013.

[22] Z. Yang and Y. Liu. Understanding node localizability of wireless ad hoc and sensor networks. *Mobile Computing, IEEE Transactions on*, 11(8):1249–1260, 2012.

[23] J. Yin, Q. Yang, and L. Ni. Adaptive temporal radio maps for indoor location estimation. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 85–94. IEEE, 2005.

[24] J. Zhang, P. V. Orlik, Z. Sahinoglu, A. F. Molisch, and P. Kinney. Uwb systems for wireless sensor networks. *Proceedings of the IEEE*, 97(2):313–331, 2009.

[25] B. Zhou, Q. Li, Q. Mao, W. Tu, and X. Zhang. Activity sequence-based indoor pedestrian localization using smartphones. *Human-Machine Systems, IEEE Transactions on*, 45(5):562–574, 2015.