AirLoc: Mobile Robots Assisted Indoor Localization

Chen Qiu  
Department of Computer Science and Engineering  
Michigan State University  
East Lansing, Michigan, 48824, USA  
Email: qiuchen1@cse.msu.edu

Matt W. Mutka  
Department of Computer Science and Engineering  
Michigan State University  
East Lansing, Michigan, 48824, USA  
Email: mutka@cse.msu.edu

Abstract—People carry smartphones that have a variety of radios and sensors. Increasingly, smartphone applications use the radios and sensors to determine a user’s location and to sense motion. Nevertheless, most existing smartphone applications cannot avoid accumulative errors when calculating position and movement. In this paper, we propose a novel approach, AirLoc - Adopting mobile robots to assist indoor Localization of smartphones. A moving robot employs a Bluetooth adapter and a known map to assist a smartphone to reduce its localization error. When a robot is near a smartphone, the robot sends accurate location information to users’ smartphones via Bluetooth. We design a path planning strategy for a robot to enhance the localization accuracies of smartphones over extended time periods. Moreover, in order to promote the single robot approach, we extend it to the multi-robot assisted indoor localization. The multi-robots are organized by an unbalanced tree and serve areas by the Distance/Density First Algorithm. Through experimentation and simulation in a multi-room building, we evaluate AirLoc and believe it is promising as a cost-efficient means to yield average positioning error below 0.9 meter and possibly lead to better localization results for some scenarios, including shopping mall and hospital.

Keywords—Indoor Localization; Mobile Robots; Smartphones

I. INTRODUCTION

Recent years have witnessed the proliferation of mobile devices along with the widespread use Location Based Services (LBS) [1]. Since Global Positioning System (GPS) is widely accepted and has satisfactory performance, LBS nearly covers outdoor environments pervasively. Nevertheless, GPS does not work within indoor environments. LBS are at best sporadic indoors.

Traditional indoor positioning approaches, aiming to provide LBS indoors, can be categorized to two types: device-based and device-free. Device-based indoor localization relies on special devices, such as ultrasound devices [2], [3], RFID [4], [5]. Whereas some of these systems can obtain accurate location information, the costs of the devices and their inconvenience limit their further development. For most device-free approaches [6]–[9], RSSI or Channel State Information is used to construct a fingerprint map. A fingerprint approach selects the best-matching position in the radio map as the mobile object’s position. However, building the off-line map and signal processing can be a significant challenge. In the area of robotics, Simultaneous Localization and Mapping (SLAM) [10], [11] concentrates on obtaining the robot’s own position and the map of the certain environment. Unfortunately, it does not help other devices that communicate with the robot to get their locations.

Mobile devices, especially for smartphones, feature a wide range of functionalities. Sensors equipped in a smartphone, such as a magnetometer, accelerometer, and gyroscope are used to locate users [12]–[16], [19]–[21]. These approaches do not require any special-purpose devices and off-line data training. By knowing the initial position, dead-reckoning [12] and other related approaches can track users by inertial sensor measurements over time. Nevertheless, localization errors are often caused by several factors, including inaccurate inertial sensors and the probabilistic algorithms to compute positions.

Some indoor environments, such as convention centers, museums, and hospitals provide various location services. Mobile robots may be able to supply certain services to users, such as advertisement and music. These robots have the abilities to establish their own positions and orientations within the frame of reference. By using a camera or an infrared sensor, the robot can find its route in an indoor environment and avoid dangerous situations by SLAM approaches. Besides, the robots can calibrate their positions computed by SLAM to ensure their high localization accuracies [17], [18].

Motivated by the increasing availabilities of smartphones and mobile robots, we present AirLoc(Adopting mobile robots to assist indoor Localization of smartphones), a low-cost, highly-accurate and large-scale indoor localization approach that integrates the off-the-shell smartphones with the mobile robots. By installing a tablet, the proposed mobile app and a known map on a mobile robot, the mobile robot can improve a smartphone’s localization accuracy. The robot is inexpensive, and its movement can be calibrated to have accurate position information. The robot leverages Bluetooth broadcast to send its correct location to the users’ smartphones. AirLoc provides a path for a robot so that smartphones might minimize the deviations between the ground truth and estimated positions by interaction with the robot: the robot collects Bluetooth RSSI from smartphones in different rooms. We classify different rooms into different crowd density levels by the RSSI values. Higher crowd density rooms should be served more often. By utilizing different crowd density levels, we design the Edge-Based Algorithm (EBA) to generate a robot’s moving route.

Since one robot takes a long time to travel all the rooms and the crowd density may change fast, the collected crowd density information is inaccurate. To further strengthen the performance and applicability of AirLoc, we extend the single robot approach to the multi-robot model. AirLoc uses a cloud server to store the Bluetooth RSSI submitted by multi-robots and update the crowd density levels continuously. When the mobile robots compute their serving paths, the crowd density levels can be accessed by WiFi. AirLoc organizes multi-robots by an unbalanced tree, which is dynamic and low complexity. In each layer of the unbalanced tree, the robots are divided into two sub-groups. The sub-group containing more robots serves...
the area with higher crowd density. In the base case of a serving tree, each robot obtain its serving area by the Distance/Density First Algorithm (DDFA) and serve it via EBA.

In our field study and large scale simulation, AirLoc improves the smartphones’ localization results successfully with low cost and acceptable complexity. This approach can be potentially applied in more indoor buildings to provide practical applications, such as in-store navigation, location-based healthcare.

**Key Contributions**

1) To the best of our knowledge, AirLoc is the first of it kind to i) use mobile robots to interact with smartphones to help indoor localization; ii) design and evaluate a system to organize multi-robots for improving the smartphones’ positioning information in real indoor environments.

2) To apply AirLoc on large scale indoor environments, based on the crowd density distribution, we design an algorithm to generate the optimized serving route for a single robot. AirLoc exploits an unbalanced tree and Distance/Density First Algorithm (DDFA) to deploy the multi-robots with low complexity and costs.

3) AirLoc updates the crowd density levels continuously to further increase the smartphones’ localization accuracies.

The rest of the paper is organized as follow: preliminaries are introduced in section II. Single robot-assisted indoor localization is presented in section III. We extend the design to multi-robot in section IV. System evaluations are shown in section V. Related work and performance comparison are discussed in section VI. Section VII provides the conclusion and future work.

**II. PRELIMINARIES**

Users of smartphones can obtain location services without any extra devices. They employ sensors (accelerometer, gyroscope) on the smartphone to compute motion traces in certain environments. However, for some sensors used for localization, such as UM6, UM6-LT [22], small errors in the orientation estimation will cause serious deviations in the measured acceleration, velocity and position. The accuracy of the UM6 and the UM6-LT Orientation Sensors are expected to be within about 2 degrees. Taking 2 degrees as an upper bound, when UM6 or the UM6-LT are leveraged for velocity and position estimation, 10 seconds will accumulate 3.4m/s of velocity error and 34.2m of positioning error.

Mobile robots have been used in some research or commercial areas and can interact with smartphones via wireless communication. By sending correct location information to smartphones, mobile robots boost the positioning accuracies of users’ smartphones. Namely, the location information on smartphones are calibrated while users are communicating with the mobile robots. To illustrate this perspective, we conduct preliminary evaluation in a small and empty environment, the size of the environment is $4 \times 4m^2$. We simulate a single smartphone’s motion and its interaction with a robot. The smartphone moves randomly and the robot moves on the diagonal line of the environment.

On the head of the robot, an Android tablet installs an application and a known map. The robot gets accurate location information by the map and calibrates its own position. For users of smartphones, a third-party application on the smartphone draws the user’s trajectory by acceleration and direction. The acceleration and direction are obtained from the accelerometers and magnetic sensors. For most motions of people who carry a smartphone, their accelerations are between $0-20m/s^2$ [23]. We consider the initial acceleration $a_x$ (on $x$ axis) to be $0.04m/s^2$, the initial acceleration $a_y$ (on $y$ axis) to be $0.10m/s^2$, and the observing time period from 1s to 40s. In the beginning, the acceleration values obtained from the accelerometer are incorrect. The margin of acceleration error on the $x$ axis and the $y$ axis is 50%. The trajectory generated on a smartphone deviates from the ground truth.

Common smartphones and tablets have Bluetooth adapters. As a low energy technology, Bluetooth works within a short range. Received Signal Strength Indicator (RSSI) is a measurement of the power present in a received radio signal. Different from Channel State Information, RSSI values can be accessed on most modern mobile devices. RSSI is in units of “dBm” (dB per milliwatt). The smaller magnitude negative numbers denote to the higher signal strength. For the percentage of RSSI, the expression to convert is: $\text{rssiPercentage} = (\text{currentRSSI} / \text{RSSIMAX}) \times 100$. The RSSI value is influenced by the distance between the sender and receiver of the radio frequency signal. Shorter distance represents stronger RSSI [24].

The relationship between distance and Bluetooth RSSI values is used to calibrate the deviations: the robot sends its accurate position to smartphones when they are close (distance between them is less than 1.5 meters, the Bluetooth RSSI value obtained from smartphone is no less than 80%). After we started our simulation, the robot sends the accurate location

![Diagram of AirLoc](image-url)
message to the smartphone after 20 seconds, then, for the user’s smartphone, the “new” received position from the robot replaces the “old” position. The sending position message lessens the deviation of the computed position on the smartphone. Then, we repeat the simulation 50 times. The results are shown in Figure 2. The blue line refers to the distance from the computed position to the ground truth without any calibration. The red line is the distance after calibration. The shadow areas denote the confidence interval for each case. Less distance means more accurate localization. It indicates: 1) as time increases, the deviation also increases; 2) the error range after position replacement is reduced.

Then, by employing the same experimental environment and devices, we record the accumulative deviation distance (Euclidean Distance) from the ground truth for each second. We control whether and when the robot sends the location information. Sending frequency is defined as the number of the location messages sent to the smartphone per hour. As Figure 3, we test the sending frequency from 0 to 100. The y axis indicates the smartphone’s accumulative deviations after our simulation. We conclude: 1) by sending accurate location messages to the smartphone, the robot can reduce the smartphone’s localization error, and 2) if the smartphone receives accurate location messages more frequently, the accumulative deviation decreases more. Therefore, we need to design an approach that the robot may frequently send its accurate location messages to the smartphone.

a shorter distance. If the detected signal strength on a robot is above a certain threshold, distances between the Bluetooth adapter (on a robot) and receiver (on a smartphone) can be ignored. The robot can send its accurate position from an installed map to a smartphone. The location from robot replaces the location calculated by the smartphone. When the installed map and routes generated from the robot are accurate, the localization accuracy on the smartphone will increase.

Next, we attempt to provide a serving route for the robot. The objective of design is to minimize the deviations of customers’ smartphones. As analyzed in our preliminaries, if the robot has more chances to send the accurate location messages to smartphones, the deviations of smartphones can be reduced more often. Hence, the robot should distinguish which rooms have more users.

Crowd density is introduced to measure the number of people in a certain area. In addition to broadcasting position information to a smartphone, a robot can collect such data in a certain room: 1) the number of discovered smartphones, 2) the mean value of Bluetooth RSSI, the two types of data can reflect the crowd density for the room [6], [8]. AirLoc adopts such two types of data as features for clustering. As Figure 5, after collecting these data in each room, k-means clustering algorithm divides crowd densities in different rooms into different ranks. Based upon crowd densities, we will assign corresponding serving time periods for different rooms. In higher crowd density rooms, the robot will spend more time to broadcast location messages. In lower density rooms, the robot will serve less time.

III. SINGLE ROBOT ASSISTED INDOOR LOCALIZATION

Figure 4 shows an overview of the single robot assisted indoor localization. By installing a tablet computer on the mobile robot, the robot contains the proposed application and a known map. The moving robot in our approach has two parallel functions: 1) send location messages to a smartphone, and 2) collect crowd density information in rooms. The robot employs Bluetooth broadcast to communicate with the smartphones.

Since the Bluetooth signal strength depends on the distances between the adapter and the receiver, the stronger signal means...
TABLE I: Notions in AirLoc

<table>
<thead>
<tr>
<th>Terms</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>N, n, P, i, j</td>
<td>Number of serving rounds, rooms, robots, current position, next position</td>
</tr>
<tr>
<td>V, V̅</td>
<td>Set of rooms, set of visited rooms</td>
</tr>
<tr>
<td>G[V − V̅]</td>
<td>Sequence of rooms to be served</td>
</tr>
<tr>
<td>d(i,j)</td>
<td>Time from moving room i to j</td>
</tr>
<tr>
<td>Dist(i,j)</td>
<td>Euclidean distance from room i to j</td>
</tr>
<tr>
<td>Sj</td>
<td>Possible serving time for room j</td>
</tr>
<tr>
<td>Tk(V̇, j)</td>
<td>Time cost for serving round k</td>
</tr>
<tr>
<td>Den(i)</td>
<td>Crowd density level for room i</td>
</tr>
<tr>
<td>α, β</td>
<td>Two separate serving areas</td>
</tr>
<tr>
<td>e0, eβ</td>
<td>Two initial positions of α and β</td>
</tr>
<tr>
<td>T1, T2</td>
<td>Thresholds for constraining merging samples</td>
</tr>
<tr>
<td>R</td>
<td>Radius of increasing serving area</td>
</tr>
<tr>
<td>HDA, LDA</td>
<td>High Density Area, Low Density Area</td>
</tr>
<tr>
<td>nHDA, nLDA</td>
<td>Number of mobile devices in HDA, LDA</td>
</tr>
<tr>
<td>ω</td>
<td>Parameter for evaluating room’s crowd density</td>
</tr>
<tr>
<td>θ</td>
<td>Parameter for assigning robots</td>
</tr>
<tr>
<td>nvi</td>
<td>Number of scanned devices in room i</td>
</tr>
</tbody>
</table>

Algorithm 1 Edge-Based Algorithm (EBA)

Input:
N, i, j, V̇, V̅, d(i,j), G[V − V̅];

Output:
Tk(V̇, j); G[V − V̅];

1: for each N > 0 do
2:     robot starts serving for G[V − V̅];
3:     traveling & scanning & sending message;
4:     V̇ ← lowest density rooms by k-means(V);  
5:     // delete nodes and joint edges with lowest density level;
6:     G[V − V̅] ← G[V − V̅] − V;  
7:     // Recursive relation for computing time cost
8:     Tk(V̇, j) = min(Tk−1(V̇ − j, i) + d(i,j));
9:     V̇ = i ∪ V̇, K = K − 1;
10:    for γ = num(G[V − V̅]); γ > 0; γ - - do
11:         assigning the same serving time for v ∈ G[V − V̅];
12: end for
13: end for

Before generating the serving paths for a mobile robot, we assume the robot moves between different rooms at uniform speed so the time to travel between different rooms can be calculated by distance (time cost on path is proportional to distance). The real map is converted to an abstract graph: rooms can be abstracted as nodes; paths between different rooms can be seen as edges. A robot’s optimal serving route is the route that a robot can minimize the smartphones’ deviations from the ground truth in a certain time period. Because the movement of crowd is difficult to predict, it is difficult for a robot to obtain the optimal route. Nevertheless, we design an algorithm to generate serving routes with close serving results as optimal routes.

Edge-Based Algorithm (EBA): Edge-Based Algorithm aims to find an optimal route covering each room and reduce time costs. It is derived from the TSP (Traveling Salesman Problem). TSP does not consider the time cost in each node. However, in AirLoc, the time length of serving in each room is crucial. For EBA, each room will be assigned the same serving time in each serving round. The rooms with higher crowd density are served more often than lower density rooms. As Figure 6, Room 1 (R1 for short), R4 are high density rooms, R2 is medium density room, R3 and R5 are low density rooms. We specify the robot to go around the whole map 3 times. High density rooms will be visited 3 times, medium density room will be visited 2 times, low density room will be visited just once. In the first serving round, all the rooms will be visited. For the second serving round, R1, R2 and R4 will be served. For the third serving round, R2 and related edges will be deleted. Only R1 and R4 will be served. The main task in each serving round is converted to find a optimal tour in the remaining map. It is similar to TSP problem, which could also be solved by dynamic programming in Algorithm 1. The total time complexity of EBA is \( \Theta(n^22^n) \).

IV. MULTI-ROBOT ASSISTED INDOOR LOCALIZATION

A. Single is not enough

Although the single robot approach is effective for some indoor environments, the problem is more challenging if 1) robot assisted indoor localization approach can be applied in more types of indoor environments, especially for the environments containing more rooms; 2) achieves higher positioning accuracies for smartphones.

The complexity of the EBA is sensitive to the number of rooms. If we adopt some common tablets (such as Google Nexus 7, iPAD, the memory is within 4G), the single robot approach can handle a map within 1-30 rooms. In order to extend the existing approach to a larger indoor environment, a new approach that can process the map with more rooms should be provided.

When a single robot employs crowd density to generate routes, how to guarantee the accuracy of crowd density estimation is intractable, because 1) people do not always keep stationary, the crowd density distribution changes at anytime, the samples collected in each room need to be updated in time. Unfortunately, one robot cannot cover many rooms in a short time; 2) the crowd density levels used in EBA reflect the crowd distribution of previous periods rather than the current period.

B. Two Robots Working Model

1) Graph Division Strategy: Before employing multi-robots to construct our system, we consider how two robots work together first. AirLoc tries to partition the whole graph to two components. Each robot will serve one of them by EBA. The strategy of graph partition should satisfy: 1) the robots allocate
more time to serve higher density rooms as possible, and 2) limit the time costs on the edges.

We propose the partition strategy in Fig. 7(a)-7(b). We abstract a two dimension plane. The samples on the plane represent the rooms on the map. The number of each sample is the crowd density level of the room. The density level 10 is the highest level and 0 is the lowest level. The samples’ density levels are obtained by accessing the cloud server. The cloud server computes the crowd density levels for each room by the k-means algorithm. Two rooms (0 and 8) are chosen as the initial center of the serving area. By increasing the radius $R$ of the area, rooms that are close to the initial samples are merged into the two serving areas by iteration. This merging procedure is based on the Euclidean distance. It is named "Distance First Algorithm".

In an indoor building, some rooms close to each other might have different crowd densities. For example, in Fig. 7(a), one room with density level 1 is merged into the high density area by the Distance First Algorithm. To tackle the problem, in contrast to the Distance First Algorithm, Fig. 7(b) depicts that samples are merged into the two areas by closest density levels rather than distances, it is named "Density First Algorithm".

AirLoc takes advantage of the above two algorithms. We define the threshold $T_1$ and $T_2$ as the constraint to make a balance between "density first" and "distance first". $T_1$ refers to the difference between the crowd density level of initial room and the crowd density level of the room being merged. $T_2$ is the difference between the crowd density level of the room merged in previous iteration and the crowd density level of the room being merged. If $T_1$ and $T_2$ are set as the larger numbers, it means even if the two samples’ density levels have significant differences, they can be merged in one serving area when their positions are close to each other. If $T_1$ and $T_2$ are set as the smaller numbers, when the two samples’ density levels are different, they have less chances of being merged. This strategy is named "Distance/Density First Algorithm (DDFA)".

The complexity of DDFA is $O(n^{2k}log n)$, $k$ refers to the number of robots, $n$ is the number of rooms. Whereas this algorithm is limited by the number of robots, it is not sensitive to the number of rooms. For most common tablets, which have more than 1G memory, they can handle 100 rooms if they use DDFA. We can conclude: 1) DDFA can be used on the map more than 100 rooms; 2) DDFA is difficult to be implemented on the group containing more than 4 robots, thus, it is necessary to extend our approach from 2 robots to $n$ robots ($n$ is greater than 2); 3) after obtaining their serving areas by DDFA, the robots will adopt EBA to travel on the assigned serving area. Since each serving area does not include many rooms, the memory bottleneck will not occur.

2) Preemption: By applying DDFA, the two robots have obtained their serving areas respectively. The area containing more higher density rooms is High Density Area (HDA), the other with lower density rooms is named Low Density Area (LDA). Since the robot travels on LDA is not as efficient as HDA, if the number of scanned smartphones in LDA is less than 10% of scanned smartphones in HDA, the robot in LDA will go to HDA to serve. This mechanism can be seen as the HDA preempting the serving period of LDA. The time of initial preemption period is $21\%$. When the LDA robot moves back to LDA, if the number of scanned smartphones in LDA is still less than 10% of scanned smartphones in HDA, the time of the second preemption period next will increase to $22\%$. Therefore, the time of $n$th preemption period is $2^n\%$. When the ratio is greater than 10%, the preemption will end and the exponent will decrease to 1.

C. Extension to Multi-robot

In order to extend the existing approach to the multi-robot model, AirLoc assigns more than 2 robots to different serving areas. If we generate serving areas for multi-robots by DDFA directly, the computational complexity might become an obstruction for common tablets. Hence, we partition the robots to serving groups layer by layer as a tree. Before reaching the bottom layer, for each layer, we focus on how to divide the robots into sub-working groups. One group is separated into two sub-groups. Each sub-group has the same number of robots. It can be seen as two robots and using the DDFA to generate two serving areas, with each subgroup responsible for one of them. Then, each sub-group is divided again until it reaches the base case (bottom) of the tree. This tree is a balanced tree.

Nevertheless, this approach has a shortcoming: even though one serving area has low density distribution and the other’s density distribution is high, the number of robots serving them is the same. It contradicts our goal to allocate more robots to serve high density areas.
Algorithm 4 Generating Unbalance Tree (GUT)

1: Input: \( P \) mobile robots, original graph; Output: unbalance serving tree;
2: if not reach the base case then
3: split the graph into two sub-graphs by DDFA;
4: find HDA, LDA by computing \( \omega \) by equation (1);
5: allocate \((P \times \theta)/(\theta + 1)\) robots to HDA;
6: allocate \( P - (P \times \theta)/(\theta + 1)\) robots to LDA;
7: for in each split sub-graph \( g \) do
8: call GUT(\( g \));
9: end for
10: else
11: call Edge Based Algorithm;
12: end if

AirLoc proposes an unbalanced tree model to address the problem. First, an alternative method to measure the crowd density of serving area is introduced: as Fig. 7(d), by building an \( x-y \) plane, the \( x \)-axis refers to the number of mobile devices and the \( y \)-axis refers to the average RSSI value. We define the parameter \( \omega \) to depict the crowd density for each room, the \( \omega \) density area is introduced: as Fig. 7(d), by building an \( x-y \) plane, the \( x \)-axis refers to the number of mobile devices and the \( y \)-axis refers to the average RSSI value. The shadow area on the plane represents the value of \( \omega \) for a room. Larger size of the shadow area means higher crowd density. As equation (1), \( i \) is the \( i \)th room, \( nd_i \) indicates the number of devices, \( m \) refers to the number of smartphones in room \( i \), \( RSSI_i \) is the signal strength received from smartphone \( j \). \( \omega_i \) is the \( \omega \) value of room \( i \).

\[
\omega_i = (nd_i \times \sum_{j=0}^{m} RSSI_j)/m, \theta = (\sum_{i=1}^{H} \omega_i)/(\sum_{j=1}^{L} \omega_j)
\]

where \( H \) represents the number of rooms in higher crowd density area, \( L \) is the number of rooms in the lower crowd density area. If \( \theta \) is a large number, it means the higher crowd density area needs more robots to serve. Based on \( \theta \), when dividing robots into sub-groups, we allocate the number of robots as follows: let \( P \) denote the number of robots, the higher crowd density area will be assigned \((P \times \theta)/(\theta + 1)\) robots, the rest of robots will be sent to the lower crowd density area. If \((P \times \theta)/(\theta + 1)\) is not an integer, it can be processed as the ceiling of \((P \times \theta)/(\theta + 1)\). Therefore, if there exists different crowd density distributions in the two serving areas, the serving tree will be formed as an unbalanced one.

D. Dynamic Return

We explained the mechanisms for sending robots to a serving area, but we did not mention how the robots go back to the initial position. For a multi-robot system, the time costs of returning might be large. Thus, the paths for returning are important. A practical and concise method is designed for generating the returning paths: find a new tree root that is relatively close to each leaf node. Namely, we find the node \( k \), which has the smallest sum of distances between \( k \) and each other room \( i \), then, we arrange \( k \) as the “new” root. After finishing the tasks in one round, all the robots will move to the new root and restart the next serving round.

V. EVALUATION

A. Experimental Setup

As illustrated in Figures 8-9, we performed our evaluation on the first floor of Engineering Building at Michigan State University, which is an indoor environment containing more than 130 rooms. Rooms are abstracted as nodes, corridors or paths that are narrower than 3 meters are abstracted as edges. The corridors or paths whose width are greater than 3 meters also processed as nodes.

In our indoor experiment, we consider the TurtleBot [17] as the initial mobile platform, it is a common type of robot with multifunction and decent price. The height of the tablet (to be installed on a robot) is 1 meter, which is similar to the height of user’s pocket. The speed of is 0.3m/s. There are 0-6 volunteers in each room or hallway, each volunteer carries Samsung Galaxy 4 smartphone or Google Nexus Tablet and turn on the Bluetooth. Volunteers in the experimental environment walk freely.

By running proposed program on a tablet and smartphones, for each 10 seconds, the tablet scans other Bluetooth devices and collects the RSSI values and the number of discovered devices periodically. Fig. 8(a) shows the scenario that we conduct the experiment. Based on the experiment, for each smartphone, it receives the location messages from the robot via Bluetooth communication. Fig. 8(b) and Fig. 8(c) demonstrate how a robot moves and collects data in a room or hallway. GN refers to the RSSI obtained from the user’s Google Nexus tablet; SG refers to the RSSI obtained from the user’s Samsung Galaxy smartphone.

On the completion of collecting data in one robot experiment, we extend our indoor experiment results to a large scale simulation. By using the same floor plan, our simulation includes 16 mobile robots that carry Google Nexus 7 tablets to visit different places in the building. Each robot travels and works by AirLoc. The 350 volunteers are distributed in each room or hallway rather than the rooms where the single robot collected Bluetooth information. After multi-robots collect samples of Bluetooth RSSI values in the first round, we use
There are three parameters in the dataset built by collected samples: number of devices, mean value of RSSI, and identifier (ID) of each room. As shown in Fig. 11(d), collected samples are distributed on the two-dimensional surface. It is formed by two features: average value of RSSI and the number of smartphones. After executing the k-means clustering algorithm, each room is categorized to the corresponding crowd density level. The size of each grid is 0.4 × 0.4m.

In evaluation phase, we seek to answer several questions, including: (1) whether AirLoc can increase smartphones’ localization accuracies; (2) how well is the updated crowd density levels of AirLoc; (3) how some features such as number of robots can influence the performance of system; (4) whether proposed technologies such as unbalanced tree, dynamic return can make contribution to AirLoc system, how they enhance the performance of AirLoc.

### B. Metric of Evaluation

Besides Euclidean distance, to measure the localization accuracies of AirLoc, we introduce other metrics:

One metric is $E = \sum_{i=1}^{N} D_i$, let $N$ denote the number of smartphones; $D_i$ refers to the deviations grids for each smartphone; $E$ refers to the remaining errors after carrying out the proposed approaches; The size of each grid is $0.4m \times 0.4m$.

The other metric is Location Entropy. The expression of Location Entropy is: $L(x) = -\sum_{i=1}^{m} P(x_i) \log_2 P(x_i))$. Higher value of the location entropy represents the localization results of all the smartphones deviate more from the ground truth. $m$ is the number of grids; for each grid $P(x_i)$ represents the probability that the smartphone’s estimated position is the ground truth. $P(x_i)$ equals to the ratio of (times of estimate successfully)/(estimated times). In our simulation, we record the $P(x_i)$ of each grid every 5 seconds.

### C. Evaluation of Crowd Density Updating

In indoor environments, such as convention center, hospital, and hotel, the crowd density distribution is always changing. For the single robot approach, the robot computes and updates the crowd density levels after each serving round. As the reasons explained in the previous section, it is difficult to guarantee the correctness of crowd density levels. Fig. 11(a) illustrates the reasons for crowd density variations.

For each serving period in multi-robot system, we define three serving slots as Fig. 10: 1) T Slot (Tree generating Slot): the robots build the unbalanced tree until they are assigned to the final serving areas; 2) S Slot (Serving Slot): when the tree reaches bottom (base case), they will conduct EBA that are also relied on the latest crowd density; 3) R Slot (Return Slot): after each period of serving, all the robot return to the “new” robot.

The multi-robot system can provide more accurate crowd density: 1) using global and concurrent Bluetooth information to replace the Bluetooth information collected serially by the single robot, the computed crowd density levels will be closer to the ground truth; 2) before each robot traveling on serving area, it can access the latest crowd density levels to help them make correct decision by proposed algorithms.

Then, we compare the location entropies of AirLoc when it adopts static or updated crowd density information. Fig.
11(b) plots the location entropy values on the y axis, that are the average location entropy values when each round ends. Static crowd density refers to robots that use the crowd density obtained at the end of first round to make a decision: OPOS (One Period One Sample) represents using the crowd density levels obtained at the first splitting group in each round when generating the serving tree. Namely, for one period, as T Slot, all robots do not update the crowd density levels when they execute the proposed algorithm until the next period. OPMS (One Period Multi-Sample) refers to the crowd density levels obtained each 10 seconds when the system generates the serving tree, forming room clusters and using EBA. By employing updated crowd density, AirLoc reduces the deviations and increases the localization accuracies gradually. Then, we repeat this simulation 100 times, shown in Fig. 11(e), we leverage the number of deviation grids to illustrate the advantages of using the crowd density information that is updated continuously by multi-robots. It proves that using concurrent crowd density information collected by multi-robots can reduce the deviations effectively. The shadow areas in Fig. 11(e) represent the confidence intervals of the simulation.

D. Evaluation of Reducing Deviation

Keeping the same conditions as the previous experiment, as Fig. 9, we study the case of each user. Even if one user’s estimated trace deviates sometimes, after receiving the accurate location messages from mobile robots, the user’s estimated trace is close to the actual one. Next, we compare the 1) the robots using dynamic return versus static return, 2) Balanced Tree and the Unbalanced Tree mechanisms. Fig. 11(c) shows that: 1) Unbalanced Tree has better performance than Balanced Tree; 2) Dynamic Return has less deviations than returning to a fixed root. Both Unbalanced Tree and Dynamic Return technologies can help the smartphones’ localization.

Fig. 11(f) illustrates the two relationships: one is the relationship between localization errors and average degree of nodes, the other is the relationship between deviations and number of robots. The degree of a node represents the number of edges connected to the node. Higher degree means more edges and better connectivity. For the single robot model, it employs EBA to serve all the rooms. For multi-robot approach, it uses OPMS and EBA. The location entropy is obtained after 10 serving rounds. We draw the conclude: 1) proposed approaches work well, and 2) when the average degree increases, the remaining deviations from the ground truth will decrease, because the better connectivity gives the robots more opportunities to choose better optimized serving routes, 3) more robots in AirLoc further enhance smartphones’ localization. In general, when we deploy no less than 8 robots and run AirLoc over 8 rounds, the average localization error of each smartphone is not beyond 0.81m.

VI. RELATED WORK AND PERFORMANCE COMPARISON

Device-based approach: Cricket [3] is a typical device-based systems. As Cricket, it has location beacons attached to the ceiling of a building and receivers. People carry receivers to obtain RF signal transmitted from the location beacons on the ceiling periodically. Cricket can locate the position of a user within 3cm. Although people often carry smartphones, the current smartphones do not integrate these devices.

Device-free approach: In RADAR [7], one of classical device-free method, multiple WiFi Access Points (APs) are heard at each location. Some stations provide overlapping coverage of the area of interest. RADAR makes use of a signal propagation model to estimate the object location to a great accuracy. In fact, for most of device-free approaches, they can only achieve room-level accuracy. In addition, training the radio-map and signal analysis are labor-intensive and time-consuming.

SLAM: robots use SLAM (Simultaneous localization and mapping) [10], [11] to create a new map within an unknown environment, or to update a map within a known environment. Also, the robots can compute and calibrate their current location information. For example, Turtlebot [17], one of the common robots, has two application (gmapping and amcl) to help it find a reasonable route in one room and compute correct positions [18]. But SLAM highly relies on the sensors, such as laser or camera. Besides, current SLAM approaches, the robots can only compute location information for themselves, other mobile devices (as smartphones, PDA) cannot obtain location information by interacting with the robots.

Smartphone approach: Dead-reckoning [12] calculates
TABLE II: Comparison AirLoc with other indoor localization systems

<table>
<thead>
<tr>
<th>Approach</th>
<th>Signal Type</th>
<th>Accuracy</th>
<th>Cost</th>
<th>Device-Based</th>
<th>Scalability</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>AirLoc</td>
<td>Bluetooth, WiFi</td>
<td>within 1m</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>High scalability and accuracy. No extra device, Low cost</td>
<td>Need known map</td>
</tr>
<tr>
<td>Ultrasound assistant</td>
<td>Ultrasound</td>
<td>10cm to 1m</td>
<td>High</td>
<td>High</td>
<td>Limited</td>
<td>High accuracy</td>
<td>Need expensive devices for each user</td>
</tr>
<tr>
<td>FingerPrinting</td>
<td>WiFi, cellular and RF signal, Bluetooth, FM</td>
<td>Room Level</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>Device-Free, Low cost</td>
<td>Complex training, Low accuracy</td>
</tr>
<tr>
<td>SLAM</td>
<td>Laser, WiFi</td>
<td>within 5m</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
<td>Accuracy is fine, Self-calibrate position</td>
<td>Just serving the robot, Need special sensors</td>
</tr>
<tr>
<td>Smartphone Based</td>
<td>WiFi</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>No extra device for customers, Low cost</td>
<td>Error accumulation, Low accuracy</td>
</tr>
</tbody>
</table>

people’s current positions by using a previously determined position, but errors accumulated from inertial sensors cannot be avoided. UnLoc [13] improves the accuracy by using a virtual landmark. Sometimes, the direction obtained from acceleration is not equal to people’s direction due to the manner that people carry the smartphone. The result of localization is still not highly-accurate. Guoguo [19] and SurroundSense [20] combine attributes such as optical and acoustic collected by inertial sensors to obtain correct locations, but filtering the noise data obtained from sensors and data off-line training are still challenging. Zee [15] and SAR [16] have the bottleneck of establishing the mapping between the physical environment and wireless signals.

For AirLoc, Table II shows a comparison with the state-of-the-art indoor localization solutions. AirLoc does not request smartphones’ users carrying extra devices. Different from most of other approaches, AirLoc can be deployed in the large environment with low cost. Besides, existing smartphone-based approaches are room levels or have serious error accumulation, even if Bluetooth communication has its range limitation (10 meters) and the people’s crowd distribution is dynamic, after conducting serving rounds iteratively, AirLoc promotes these approaches to achieve better localization results.

VII. CONCLUSION AND FUTURE WORK

Most of previous indoor localization approaches cannot avoid accumulative errors. AirLoc boosts smartphones’ localization accuracies on large indoor environments. We build an unbalanced tree to organize the multi-robot. From the root to bottom, the robots are divided into sub-groups according to the distribution of people’s crowd density. At the bottom, all the robots are assigned their working areas by DDFD and serve these areas by EBA. Evaluation results indicate 1) AirLoc can be applied on the map containing more than 100 rooms; 2) localization errors of smartphones are reduced effectively via AirLoc. The average deviation of each smartphone belows 0.9m; 3) the time complexity of introduced algorithms are acceptable for common tablets.

We have applied AirLoc in a hall with multiple rooms. In our future work, deploying mobile robots in various scenarios will help us to better analyze the proposed approaches. Besides, with known accurate location information, more location based applications will be supported by AirLoc, such as the smart navigation in a shopping center.

ACKNOWLEDGMENT

This paper was supported in part by the National Science Foundation grant no. CNS–1320561.

REFERENCES