iFrame: Dynamic Indoor Map Construction through Automatic Mobile Sensing

Conference Paper · March 2016

DOI: 10.1109/PERCOM.2016.7456500

2 authors, including:

Chen Qiu
Michigan State University
10 PUBLICATIONS **114** CITATIONS

SEE PROFILE

iFrame: Dynamic Indoor Map Construction through Automatic Mobile Sensing

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*—Many pervasive computing applications depend upon maps for navigation and support of location based services. Maps are commonly available for outdoor pervasive applications from a variety of sources. An individual can determine their location outdoors on these maps via GPS. Indoor pervasive applications may also need to know the layout of rooms, doorways and hallways of buildings, and the objects and obstacles within them, however indoor maps of buildings are less prevalent. Moreover, indoor maps may need to be dynamic and updated regularly since the layout changes when objects and obstacles are added or removed by people within the building. In this paper, we present `iFrame`, a dynamic approach that leverages existing mobile sensing capabilities for constructing indoor floor plans. We explore how `iFrame` users may collaborate and contribute to constructing 2-dimensional indoor maps by merely carrying smartphones or other mobile devices, and to allow their mobile devices to share information with other users' devices. The `iFrame` approach consists of four steps: 1) Abstract the unknown indoor map as a matrix; 2) Leverage collaborating mobile devices that incorporate three mobile sensing technologies - accelerometers to support dead reckoning, Bluetooth RSSI detection, and WiFi RSSI detection; 3) Combine the three methods by Curve Fit Fusion (CFF), and 4) Extend `iFrame` from one room to a whole building by shadow rates and anchor points analysis. We conducted a deployment study that shows `iFrame` is a light-weight and unattended approach that provides a skeleton map of a real building effectively and automatically. The layouts of 12 rooms are reconstructed within 5-10 minutes. Changes of layout in indoor maps can be detected and the resolution of the reconstructed indoor floor plans can be improved when there is an increase in the number of cooperating users.**

I. Introduction

Map availability is essential for Location Based Services [1], such as tracking and localization services. In outdoor environments, Google Map, Yahoo Map and other services provide outdoor maps for the users of mobile devices. Unfortunately, maps are much less commonly available for indoor environments. In order to provide indoor floor plans, many people may need to dedicate significant time to construct the floor plans and make them available to commercial channels. Google Indoor Map [2] has collected over 1,000 indoor floor plans of airports and shopping malls in U.S. and Japan. These maps are built manually, which may be tedious and high cost. Moreover, even if the indoor maps are constructed, the layouts within these buildings may often change. For example, furniture within one room might be moved. This leads to our question: *Can we generate a dynamic indoor map automatically and accurately without complex data training?*

Mobile sensing has been a popular research topic in recent years. Researchers use mobile devices to localize users and provide various location based services [3]–[8]. Some approaches for building indoor floor plans have been proposed. JigSaw [9] employed camera sensors to collect images. After applying point cloud algorithms, they piece together images to construct the indoor map. Although this approach can build the indoor map using smartphones, the procedure of image processing is challenging. CrowdInside [10] and MapGENIE [11] adopt dead reckoning [12] to obtain the motion traces of mobile devices users. By analysis of the characteristics of accelerations, these approaches recognize different layouts within indoor environments. Nevertheless, the accuracies of dead reckoning and the complexity of analysis need to be further improved.

In this paper we propose `iFrame`, an light-weight approach that leverages mobile sensing data to construct dynamic floor plans of complex indoor environments automatically. `iFrame` does not require commercial negotiation with providers of indoor maps. By opening the `iFrame` application on smartphones and by walking around the targeted indoor environments, users passively construct the floor plan. We abstract the indoor map as an $m \times n$ matrix. Each grid in the matrix has the same shape and area. Grids in the matrix are described by a value between two extreme states: 0 - the grid is vacant, 1- the grid is completely occupied by objects. The value between 0 and 1 represents the percentage of the area that an object occupies the grid.

Users leverage dead reckoning to obtain their motion traces. For locations a user visits that are identified by dead reckoning, the grid location will be marked vacant. Because dead reckoning may deviate from the ground truth, we seek to enhance its accuracy and therefore the proper identification of the status of grids in the matrix by employing 1) a Markov chain to calibrate the trace deviation and 2) employ Bluetooth and WiFi radios to compensate for the deviation errors. Drawing upon results from the literature, we assume the Bluetooth RSSI received from another device to be related to distance [13]. Furthermore, we suppose abrupt changes in WiFi signal strength is related to obstacle detection [14]. When a user turns on the Bluetooth option and if a connection is established to other detected Bluetooth adaptors, then we set the grids on the link between Bluetooth adapters to 0 (vacant). Since WiFi signal strength is more sensitive to noise rather than to distances, and if the WiFi signal strength decreases abruptly, then the grids that on the link are recorded as 1. Considering the imperfections of each method, Curve Fit Fusion (CFF) is introduced to combine

978-1-4673-8779-8/16/$31.00 ©2016 IEEE

the methods for improving the output matrices. In addition, we explore an approach to filter the noises caused by human crowd interferences. By executing `iFrame` iteratively in an indoor building within 5-10 minutes, we can obtain a two dimensional floor plan. Since the generated map is built by real time sensing information, even if the layouts of rooms change, the changes can be represented on the map. As numbers of `iFrame` users who share their information increases, the constructed floor plans generated from a larger data set drawn from a larger set of users will have a lower error rate and will be generated within a shorter amount of time.

The main contributions of `iFrame` are briefly summarized as follows:

1) Although some approaches have adopted crowd sourcing as a means to rebuild the indoor floor plans, `iFrame` is the first of its kind to measure RSSI values with other scanned mobile devices to help construct the layout of indoor environments.
2) We design a sensor fusion approach to combine the indoor maps computed by dead reckoning, Bluetooth, and WiFi RSSI detections. The fusion of data improves the accuracies of indoor floor plan reconstruction.
3) `iFrame` is convenient to deploy and may be used passively by smartphones' users. The mobile sensing mechanism allows `iFrame` to represent the layout changes of buildings. As more users collaborate, improved resolution and clarity of the maps are generated.

The rest of the paper is organized as follows: Section II introduces the system design. Experiments and simulations are shown in section III. Section IV provides related work. Conclusions and future work are in Section V.

II. System Design

A. Overview of Our Approach

`iFrame` leverages existing smartphone and mobile device technology without the need of complex training to construct floor plans within buildings. Before constructing an indoor map, we formulate the unknown map as a matrix. Each element in the matrix represents a grid on the map. Each grid is marked in one of two states: empty or object. As shown in Fig. 1, the procedure of map construction is composed of three phases:

1) `iFrame` is a system based on mobile sensing. Smartphone users adopt the sensing techniques to set the status of each element in matrix iteratively: i) dead reckoning, ii) Bluetooth, and iii) WiFi detections.
2) Based upon the data collected by sensors on smartphones, `iFrame` fuses the output matrices of the three methods via Curve Fit Fusion (CFF), which is supported on the cloud server. Then, `iFrame` combines the matrices built from multiple mobile devices and filters the noises caused by the *temporary shadow*.
3) `iFrame` draws the layout in each room and hallway. By introducing Anchor Points (such as entrances, stairs, and elevators), the map of a indoor building is constructed completely.

B. Map Matrix

Our task is to build the two dimension indoor floor plan. However, the initial two dimension map is unknown except for the size of the area. We divide the map into grids, and each grid is processed as an element in a matrix $\mathbb{M}$. All grids are

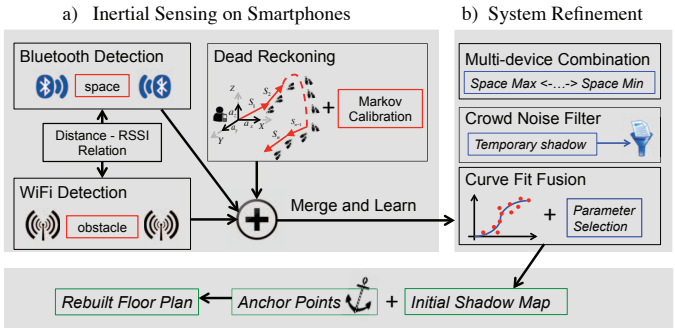


Fig. 1 System overview of iFrame.

TABLE I Main notations in `iFrame`

Terms	Definition
(x_α, y_α)	Position of users α
$RecRSSI$	RSSI values the users received
$\mathbb{M}$	Matrix of abstracted map
T_b	Threshold of Bluetooth Detection
T_{w1}, T_{w2}	Thresholds of WiFi Detection
$Dist(A, B)$	Euclidean distance between A and B
$V_{i,j}$	Shadow value of element (i,j) in M
V_{BL}	Shadow values on the Bluetooth link
V_{WL}	Shadow values on the WiFi link
M_f	Matrix generated by sensing approach f
S_f	Shadow rates computed by sensing approach f
M_{d_i}	Matrix generated by device i
$B_{m\times n}$	EBV for element $M(m,n)$ in M

squares with the same size. If a grid is vacant, it is defined as an *empty* grid. We set the value of corresponding element in the matrix to 0. If a grid is totally occupied by an object, it is defined as an *object* grid. The corresponding element in the matrix is 1. The Shadow Rate is the amount that an object partially occupies the matrix element. If the size of the grid is $1m^2$ and the objects in the grid occupy $0.7m^2$, we define the shadow rate of the grid as 0.7. $V_{i,j}$ refers the value of a element in matrix $\mathbb{M}$. The matrix $\mathbb{M}$ is as follows:

$$\mathbb{M} = \begin{bmatrix} V_{1,1} & \cdot & \cdot & V_{1,n} \\ \cdot & \cdot & & \\ \cdot & & \cdot & \\ V_{m,1} & & & V_{m,n} \end{bmatrix} \quad (1)$$

C. Dead Reckoning Detection

1) Basic Dead Reckoning Approach: When a user enters a grid on the map, the grid has empty space for the user to occupy. The grids on the user's motion traces will have their status marked as *empty* (0). Hence, the user's motion traces in a building are able to describe the layout of the building.

`iFrame` uses dead reckoning [12] to obtain the user's motion traces. The application executing on the mobile device computes movement distance in each segment by equation (2) continuously. Then, it will form the whole trace of the user's motion. a_k refers to the acceleration in period k, which is accessed from the accelerometer on a smartphone. S_k and v_k represent the related moving distance and velocity.

$$S_k = v_{k-1}t_k + \frac{1}{2}a_{k-1}{t_k}^2 \quad (2)$$

We define the generated motion trace as:

$$T = [(x_1, y_1), (x_2, y_2), .., (x_n, y_n)] \quad (3)$$

$V(x_u, y_u)$ denotes the corresponding element in matrix $\mathbb{M}$ ($1 \leq u \leq$ n).

Hence, the operation of dead reckoning detection is:

$f_D \rightarrow$ set $V(x_1, y_1), V(x_2, y_2), ..., V(x_u, y_u)$ as 0, when $V(x_1, y_1), V(x_2, y_2), ..., V(x_u, y_u)$ are on the trace T.

Unfortunately, dead reckoning has limitations: if the acceleration values from the smartphone do not reflect the human body's acceleration, for example, when the smartphone is on a user's hand, and the user shakes his/her hand, the obtained accelerations are incorrect. The generated trace will deviate from the ground truth. Furthermore, the accelerometers on most of smartphones are not highly accurate. A common sensor used for indoor localization [15] may have an error of 308 meters within one minute due to a 0.5 degree deviation occurs on the orientation sensor.

2) Trace Prediction and Calibration by Markov Chain: To improve the accuracy of dead reckoning, we proposed a trace prediction approach relying on a Markov chain. A Markov chain is a sequence of random variables X_1, X_2, ..., X_n. Given the present state, if both conditional probabilities are well defined, the future is conditionally independent of the past as formula (4).

$$P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = P(X_{n+1} = x | X_n = x_n) \quad (4)$$

In our model, Markov chain is abstracted as a directed graph:

1) Each grid in the map refers to a node, the edges in graph are labeled by the probabilities of moving from one grid at time n to the other grid at time n+1.

2) As Fig. 2, the user can only move to a neighbor grid/node from time n to time n+1. This action is represented by the transition matrix from time n to time n+1. For each grid, since the number of potential transition grids are 8, the size of transition matrix is 8×8. If the user goes to a non-neighbor node, it needs more than one step. The value k is the number of steps for transferring grids.

$P\{X(n_m + k) = j | X(n_m) = i_m\}$ is the transition probability by moving k steps at time n. It is recorded as $P_{i,j}(n, n+k)$ for short. It indicates the probability that the user is in the state/grid i at time n, and after the transition of k steps, it goes to the state/grid j. $P_{i,j}(n, n+k)$ depends on initial state/grid i, the final state/grid j, the number of steps k, instead of time n. Therefore, the transition probability of k steps can be defined as:

$$P_{i,j}(k) = P_{i,j}(n, n+k) = P\{X(n+k) = j | X(n) = i\} \quad (5)$$

When $k = 1$, $P_{i,j}(1)$ is a one step transition probability. The matrix of one step transition is recorded as $P(1)$. For n steps transition probability, the transfer probability is $P_{i,j}(n)$. The corresponding matrix is the n step transition probability matrix, named $P(n)$. By applying C-K equation [16], we can obtain:

$$P(n) = P \cdot P(n-1) = P(n-1) \cdot P => P(n) = P^n \quad (6)$$

For $P(1)$, the corresponding $P_{i,j}$ value refers to the probability of moving from state/grid i to state/grid j. Since we can count the times from i to j as $num_{i,j}$, then

$$P_{i,j} = \frac{num_{i,j}}{\sum_{j=1}^{n} num_{i,j}} (1 \leqslant i, j \leqslant n) \quad (7)$$

After obtaining the one step transfer probability matrix, we are able to compute the transition probability matrix of k steps by $P(k) = P^k$.

To predict the user's selection for the next step, it is necessary to consider both the historical and the current information of the user's traces. The information that is closer to the current time period will have more influence on the user's decision for the next step. According to this strategy, we provide the prediction formula as follow:

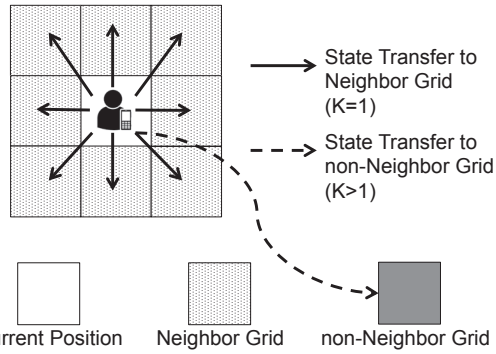


Fig. 2 Markov chain state transition. The user moves from the current grid to the neighbor grid.

$$X(t) = a_1 H(t-1) P + a_2 H(t-2) P^2 + ... + a_n H(t-n) P^n \quad (8)$$

In formula (8), t is time period of next grid, t-1 refers to the previous time period of t. $X(t)$ is the prediction probability for the next grid/state. $H(i)$ denotes the states of the next grid's previous i grids and the factors $a_1, a_2, ..., a_k$ from 1 to k are used to decide their next grids ($a_1 \geq a_2 \geq ... \geq a_k$). Then, we select the maximum element in $X(t)$, and take the maximum element's corresponding grid as the prediction grid for the next step.

When the user predicts the grids that are his/her next moving targets, if the motion trace computed by formula (2) does not include the predicted grids, the acceleration values for this time period will be replaced by the retrieved accelerations in the previous time period. Then, our system recalculates the current segment of motion trace.

D. Bluetooth Detection

Received Signal Strength Indicator (RSSI), in units of "dBm", is a measurement of the power present in a received radio signal. RSSI can be recorded from the components of most mobile devices, such as WiFi adapters and Bluetooth adapters. Bluetooth RSSI values received from the other devices are related to the distance between the devices. Shorter distances often represent stronger RSSI values [13]. We evaluated the RSSI-distance mapping relations for the Bluetooth adapters on Samsung Galaxy smartphones and Samsung Tablets. These mapping relations are pre-trained and stored as hash maps.

If two devices can detect each other, we assume that the link between the two devices has open space. We estimate the distance between the two devices from the initial map. The positions of the two devices are obtained through dead reckoning. If the corresponding RSSI values from RSSI-distance relation are close to the received RSSI values (the error range is 3dBm), namely, we assume no interference on the link. The elements V_{BL} between the link in $\mathbb{M}$ are set as 0. When the received RSSI is less than the corresponding RSSI values (beyond the error range), there might be some interference on the link. It is unreasonable to set all the values of elements on the link in M as 0. The procedure of Bluetooth detection is described in Algorithm 1.

E. WiFi Detection

Wi-Fi Direct enables devices to connect with each other without requiring a wireless access point. It also can provide

Algorithm 1 Bluetooth Detection Algorithm

Input:
$(x_\alpha, y_\alpha), (x_\beta, y_\beta)$, $RecRSSI_\alpha, RecRSSI_\beta$, $V_{BL}(x, y)$;
Output:
The latest $V_{BL}(x, y)$;
// Use the prepared Hashmap to compute
// the estimated RSSI by known positions
$Dist(\alpha, \beta) \leftarrow \sqrt{(x_\alpha - x_\beta)^2 + (y_\alpha - y_\beta)^2}$
EstRSSI $\leftarrow$ HashMap($Dist(\alpha, \beta)$);
if $(RecRSSI_\alpha + RecRSSI_\beta)/2 - EstRSSI < T_b$ **then**
  for each $V_{BL}(x, y)$ **do**
    // Once the RecRSSI is close to Estimated RSSI,
    // we add 0 for the elements on Bluetooth link
    $V_{BL}(x, y) \leftarrow V_{BL}(x, y)$ + 0;
    cnt $\leftarrow$ cnt+1 & $V_{BL}(x, y) \leftarrow V_{BL}(x, y)$ / cnt;
  end for
end if

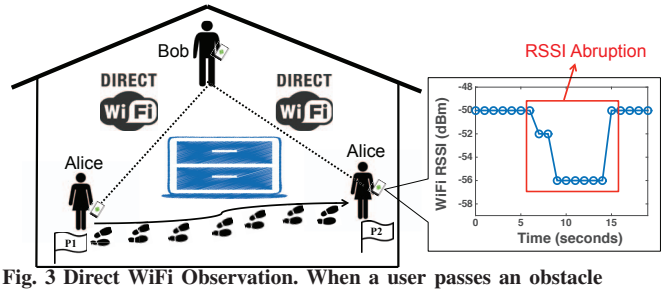


Fig. 3 Direct WiFi Observation. When a user passes an obstacle between the WiFi link, the RSSI value decreases sharply.

RSSI values for each of the connected devices. Wi-Fi Direct is supported by most current smartphones. In contrast to Bluetooth RSSI, WiFi RSSI values are not as sensitive to the distance within a short range. However, the value of WiFi RSSI is susceptible to the interference between the link [14]. It is thus desirable to uses WiFi to describe the *object* on the map. Figure 3 illustrates a preliminary observation. Alice and Bob are two user of `iFrame`. They build connections via Wi-Fi Direct first. Then, Alice walks from position 1 to position 2. On her way from position 1 to position 2, there is a cabinet between user Alice and Bob. Once Alice crosses the cabinet, the received WiFi signal strength from Bob has an obvious abrupt signal change. Based upon this phenomenon, we formulate the WiFi Detection in Algorithm 2.

In Algorithm 2, two events are pre-defined as follow:

Event (I): The differential value between the received RSSI value and the RSSI estimated by the distance is greater than the threshold T_{w1}.

Event (II): The WiFi RSSI between the two users drops abruptly (more than the threshold T_{w2}) from the previous period.

F. Matrix Fusion Mechanism

We introduced three techniques: dead reckoning, Bluetooth detection, and WiFi detection to set the values on the map matrix. However, there are some shortcomings for each of them: 1) dead reckoning can describe the user's motion traces, but the estimated positions often deviate the ground truth. For example, the accelerometer on a smartphone represents the acceleration of the smartphone rather than a human's movement. Once a user's body motion is different from the smartphone's motion, dead reckoning computation will fail due to the deviation. Though we designed the prediction mechanism to correct

Algorithm 2 WiFi Detection Algorithm

Input:
$(x_\alpha, y_\alpha), (x_\beta, y_\beta)$, $RecRSSI_\alpha, RecRSSI_\beta$, $V_{WL}(x, y)$;
Output:
The latest $V_{WL}(x, y)$;
for each time period i **do**
  $Dist(\alpha, \beta) \leftarrow \sqrt{(x_\alpha - x_\beta)^2 + (y_\alpha - y_\beta)^2}$
  EstRSSI $\leftarrow$ HashMap($Dist(\alpha, \beta)$);
  if Event (I) and Event (II) are satisfied **then**
    // Event (I):
    // $|EstRSSI - (RecRSSI_\alpha + RecRSSI_\beta)/2| > T_{w1}$
    // Event (II):
    // $|(RecRSSI_\alpha + RecRSSI_\beta)_i - (RecRSSI_\alpha$
    // $+RecRSSI_\beta)_{i-1}| > T_{w2}$
    for each $V_{WL}(x, y)$ **do**
      // When RecRSSI is close to Estimated RSSI,
      // we add 1 for the elements on WiFi link
      $V_{WL}(x, y) \leftarrow V_{WL}(x, y)$ + 1;
      cnt $\leftarrow$ cnt+1 & $V_{WL}(x, y) \leftarrow V_{WL}(x, y)$ / cnt;
    end for
  end if
end for

such errors, the deviations still occur sometimes; 2) Bluetooth detection approach may ignore some interferences that are lower than the threshold between the link; 3) WiFi detection cannot represent some vacant grids on the link that have detected objects. Therefore, the matrix computed by the three approaches includes errors. In order to refine the results, `iFrame` merges the three methods and generates improved results.

We proposed Curve Fit Fusion (CFF) to combine the matrices computed by the three methods. According to the samples from the training data, we select a proper proportion to assign different weights to the three methods.

$$\begin{cases} a + b + c = 1 \\ a \times M_D + b \times M_B + c \times M_F = M \\ a : b : c = {B_D}^{-1} : {B_B}^{-1} : {B_F}^{-1} \end{cases} \tag{9}$$

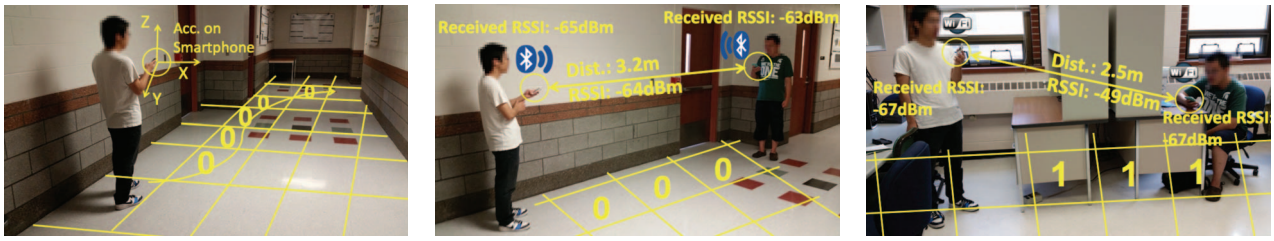
$$B_{m \times n} = \sqrt{\sum_{j=1}^{n} \sum_{i=1}^{m} (\Delta V_{ij})^2}, \Delta V = |V_g - V_e| \tag{10}$$

As given in equation (9), M_D, M_B, M_F denote the matrices computed after using dead reckoning, Bluetooth and WiFi detections. M is the matrix computed by combining the three techniques. The factors a, b, and c decide the proportion of M_D, M_B, M_F. S_D, S_B, S_F refer to the shadow rates of the three approaches in a room.

A special metric is introduced. Error of Block Value (EBV) is the error value of the estimated shadow rate in each grid. V_g refers to the real shadow rate in each grid. V_e is the shadow rate computed by our approach. If we use an $m \times n$ matrix, the Error of Block Value is defined as $B_{m \times n}$, which is the average error value of all the grids.

We collect a small-scale training dataset: by changing the layouts in three rooms, for various types of layouts, each has a different shadow rate. Then, we compute the shadow rate and error value for each case. By applying different approaches, in Figure 5, each sample on the figure is a case we tested. The figure illustrates the relation between the shadow rates and the error values for these cases.

As given in equation (11), B_D, B_B, B_F refer to the error values caused by dead reckoning, Bluetooth and WiFi detections.



(a) Dead reckoning detection. (b) Bluetooth link detection. (c) WiFi Direct link detection.

Fig. 4 Approaches for constructing the map matrix $\mathbb{M}$: (a) Dead reckoning trace determines the space areas; (b) Bluetooth link determines the space areas; (c) WiFi Direct connection detects the obstruction. To demonstrate the three approaches, the users hold smartphones in their hands. Indeed, these approaches perform regularly when the smartphones are put into users' pockets with unattended mode.

After applying the curve fit tool, we can compute the values of α and β. By knowing α and β, once we obtain the shadow rates, it is easy to calculate the proper a, b, and c values from equation (8).

$$(B_D, B_B, B_F) = (\alpha_1, \alpha_2, \alpha_3) \cdot (S_D, S_B, S_F) + (\beta_1, \beta_2, \beta_3) \quad (11)$$

G. Multi-device Combination

Since `iFrame` is a crowd sourcing mechanism, all the users of mobile devices collect and upload sensing data. Therefore, it is significant how we organize data computed from each mobile device. Our system provides three types of organizations:

1) Maximum Space: $M_{d_1} \bigcup M_{d_2} ... \bigcup M_{d_{n-1}} \bigcup M_{d_n}$
2) Minimum Space: $M_{d_1} \bigcap M_{d_2} ... \bigcap M_{d_{n-1}} \bigcap M_{d_n}$
3) Mean Value: $\sqrt{(M_{d_1}^2 + M_{d_2}^2 + ... + M_{d_{n-1}}^2 + M_{d_n}^2)/n}$

M_{d_i} denotes the matrix computed by device i. The Space Maximum represents the combination matrix that has the most space. The Space Minimum is the combination matrix that remains the least space. The Mean Value is between these two extremes. We adopt the Mean Value as the default mechanism to merging sensing data from different devices.

H. Crowd Noise Filter

One situation is non-negligible: if there exist so many users who are stationary in one room or a hallway, the generated shadow map might include some errors. The people's bodies may tend to yield some shadows in the generated map. To reduce these "temporary shadows," `iFrame` checks the motion trace of each user periodically and considers if a user does not change his/her position within 5 minutes, `iFrame` will not use his/her data until he/she leaves the position.

In addition, if many people are in a room or hallway, regardless if the people's bodies are stationary or not, there might be interference for the smartphones' radio signals. The presence of large numbers of people often cause variations of RSSI values [17], [18]. Based on this phenomenon, we need to detect the room with high crowd density and amend the variations of the RSSI caused by the presence of many human's bodies.

Three steps are taken as a heuristic solution for filtering crowd noises:

1) We divide the matrix $\mathbb{M}$ into sub-matrices from M_1 to M_n. Each keeps the same size. For M_i ($0 \leq i \leq n$), in a certain time period, we focus on three features: the number of users, the sum of variation of Bluetooth RSSI for all the devices, and the sum of variation of WiFi RSSI for all the devices.

2) By the three proposed features, we assume each M_i is a sample on a three dimensional space. The three features represent the three dimensions. By employing the k-means algorithm, we divide the sub-matrices into three types: high crowd noise level, normal crowd noise level, and low crowd noise level as Figure 6.

3) For the matrices tagged with high crowd noise level, we take actions to reduce the noise: since the matrices M_B and M_F made by Bluetooth and WiFi RSSI detections are interfered by human bodies, these data are will not be adopted. Only dead reckoning approach is acceptable for the matrices with high crowd noise level. Matrices with normal or low level crowd noise still use the three key techniques together.

I. Extend One Room to Multiple Rooms or Hallways

In order to apply `iFrame` to a real scenario, we extend our approach from one room to a building containing multiple rooms and hallways.

In a real building, some rooms or hallways have high shadow rates and others have low shadow rates. As mentioned by Curve Fit Fusion (CFF), the three detection techniques have their own features. The three techniques have different performance in rooms due to different shadow rates. Therefore, we need to assign different weights for the three methods.

We train our data by three basic types of rooms: a room with a low shadow rate, a room with a medium shadow rate, a room with a high shadow rate. We use the formula in CFF to obtain three groups of a, b, c values: one for a crowded room, one for a normal room, one for the room with few objects.

When a user enters a room or a hallway, we set the parameters a, b, c as 1/3 initially. Then, the user runs `iFrame` and computes the average shadow rate of each room for the first time period. If the shadow rate satisfies the low/normal/high shadow rate, for the next period, it will be computed by the corresponding group of a, b, and c.

J. Extend Rooms to a Real Building

1) Anchor Points Analysis: In addition to the rooms and hallways of a building, there are some special places, such as entrances, elevators, and stairs. When people walk in a building, these places should be recognized as the initial position of dead reckoning and the joints of rooms/hallways. These places are named Anchor Points. Our existing approach can detect the layout of a room and hallway, nevertheless, it is difficult to recognize Anchor Points. We employ the techniques found in CrowdInside [10] and note that the motions of users have their own acceleration ranges: 1) stationary: 0-5m/s^2; 2) elevators: 0-4.1m/s^2; 3) walking: 0-13.2m/s^2; 4) stairs: 0-20.2m/s^2. `iFrame` analyzes the types of Anchor Points via their acceleration signatures. Then, we adopt 1) correlation between the acceleration values on different axes and 2) variance of acceleration magnitude to further classify Anchor Points. According to the obtained accelerations and

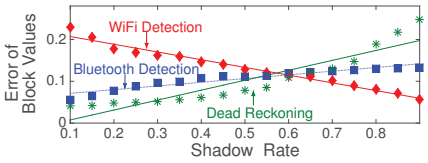


Fig. 5 Curve Fit Fusion for the sensing data collected by the three proposed methods.

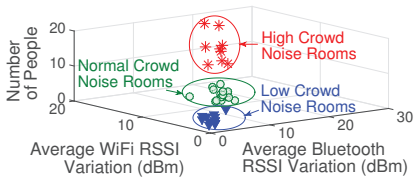


Fig. 6 Different levels of crowd noise for rooms.

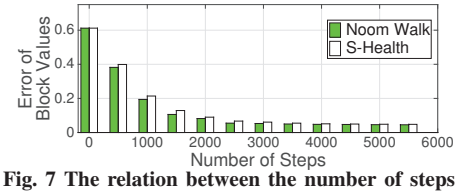


Fig. 7 The relation between the number of steps and map rebuilt accuracy.

analysis patterns proposed in CrowdInside, we identify Anchor Points to complete our map.

Although `iFrame` does not depend on the pre-installed devices (such as WiFi Access Points and Bluetooth Beacons) in a building, `iFrame` can leverage these devices to improve dead reckoning. If the pre-installed wireless devices can share their positions, once a device is close (within one grid) to a user of smartphone, the user's computed position from dead reckoning will be replaced by the device's accurate position. The distance between the wireless device and the user's smartphone is computed by the prepared RSSI-distance relations. Considering the position of the installed wireless device is known and accurate, the device also can be treated as an Anchor Point.

2) Hallway Assembling: In a building, hallways are separated by walls. Computer vision based approaches often use complex image algorithms to gather the layouts separated by walls and the walls themselves. `iFrame` solves this problem by: 1) users cannot cross the wall, therefore, dead reckoning detection will not set the related element in $\mathbb{M}$ as 0; 2) if a user in a room and a user out of a room can detect each other, WiFi and Bluetooth detections cooperate together to mark the wall related elements as 1. More samples can rebuild the wall more clearly. This method does not need extra computation and can avoid image gathering.

III. Evaluation and Discussion

A. Experiment Setup

We prototype `iFrame` on Samsung Galaxy S5 smartphones and Google Nexus 7 tablets, which support various types of inertial sensors. The version of Android is 4.4. In the experiments and simulations, users leverage Bluetooth adapters, WiFi adapters, and accelerometers. The corresponding sampling frequencies of these components are 0.2-0.06HZ, 0.25-0.5HZ, and 1HZ respectively. After turning on these sensors, the users carry the mobile devices and walk freely in the indoor scenarios. Initially, we set the shadow value of each grid as 1. For the size of each grid, if the size is too large, the accuracy of rebuilt map will be constrained. On the contrary, once the size of grid is too small, the complexity of computation will increase sharply. In our evaluation, we set the size of each grid as 0.5m$\times$0.5m.

We implement dead reckoning to generate user's motion trajectory: 1) calculate the movement direction and the motion distance in each segment and 2) merge computed segments in each time period. For Bluetooth and WiFi detections, the distance-RSSI relations for the smartphones are trained off-line and stored in hash tables. We estimate the layouts between each of connected smartphones via the proposed algorithms. Empirically derived thresholds T_b= 3dBm, T_{w1}= 4dBm, T_{w2}= 4dBm are proposed in Algorithm 1 and 2. When we combine the three sensing approaches, the initial values of a, b and c are 1/3. We adopt Mean Value pattern to merge the matrices computed from all the mobile devices.

In our evaluation, we seek to answer these questions: 1) Does `iFrame` construct the indoor layout of a building successfully? 2) Can our approach detect the change of layout of a room? 3) Can our matrix fusing mechanism improve the output results? 4) Can an increase in the number of users enhance the accuracy or speed up the rebuilding process?

B. Indoor Environment Measurement

1) Room Measurement: We conducted an experiment in the eLANS Lab of Michigan State University. As shown in Figure 8, we transform the floor plan of a room into a two dimension map described by a shadow based matrix. The gradient color (from black to white) in each grid represents the shadow state. Three people carry smartphones and walk freely in the room. After executing our approach for 10 minutes, `iFrame` constructs the indoor floor plan. With the time increasing, the estimated map more closely reflects the ground truth. For example, the generated shadow map in the 10th minute is more accurate than the shadow map in the 5th minute.

In order to verify our approach is able to represent the latest version of the indoor map, we changed the layout of the room. The proposed algorithm in `iFrame` ran continuously with unattended mode. Then, we conducted the same experiment in the changed scenario. Even if the trash cans and tables have been moved, as Figure 9, the new generated map is still close to the ground truth.

Note that when people work or live in a room, there is little possibility to walk continuously. Therefore, we did an experiment that more closely resembles people's daily life. There were three users of `iFrame` in a room. They can walk, stop, sit, and chat in the room for 1 hour.

In this case, each user adopts Noom Walk [19] and S-Health [20] Pedometer applications on their smartphones, and record the number of steps they have walked. Figure 7 depicts that even if there exists some differences for the step numbers counted by the two applications, within the increasing of the number of steps, the Errors of Block Values are reduced gradually. Then, we executed a similar experiment within 30 minutes, the experimental results are close to what we obtained in the 1 hour's group.

To analyze the effectiveness of each technique, we did the following experiments. First, we only used dead reckoning approach to build the map. Then, we added Bluetooth detection and did the experiment. In the end, we combined the three technologies together. As shown in Fig. 8 and Fig. 11(a), we conclude each technique can improve the accuracy of the map. By repeating the comparison 10 times, Fig. 11(b) provides the confidence interval for each approach and shows that our conclusion is not coincidental.

There are three participants in the above experiments to collect samples to construct the map. We observe two other control groups for 10 minutes: 1) One participant has

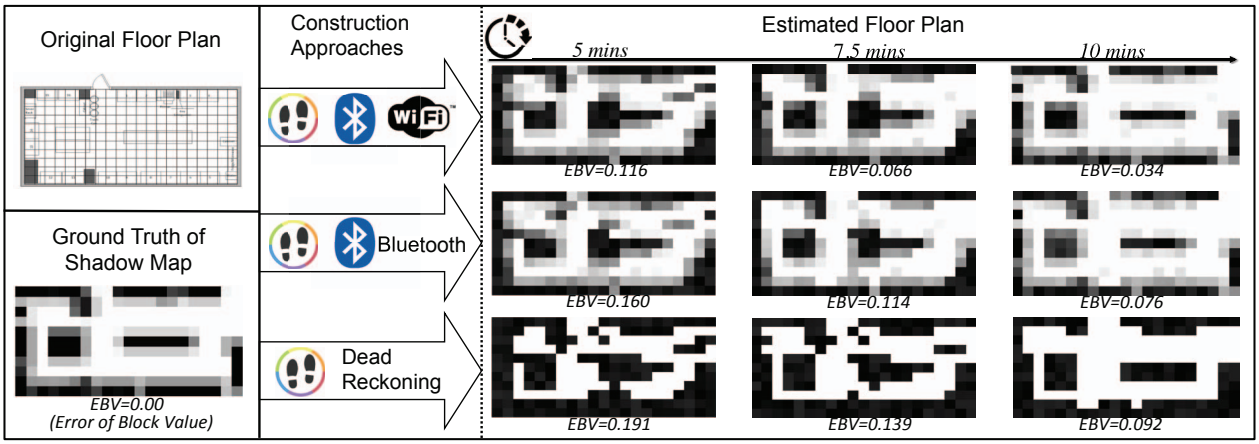


Fig. 8 The case study of indoor floor plan reconstruction. Within the time increase, the generated shadow map is closer to the ground truth. The group including the three methods outperforms other two groups.

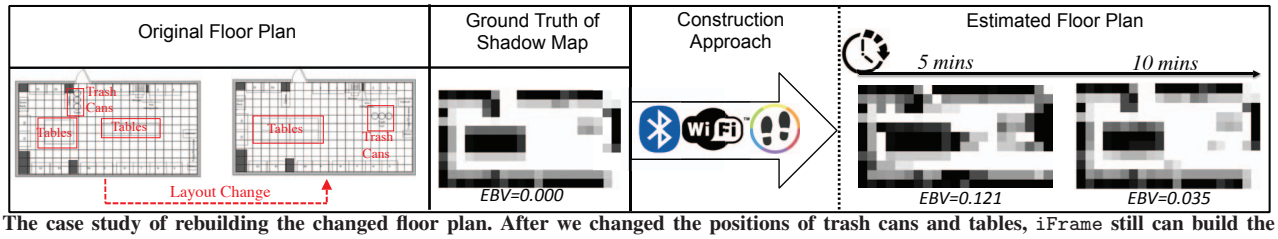


Fig. 9 The case study of rebuilding the changed floor plan. After we changed the positions of trash cans and tables, `iFrame` still can build the accurate indoor shadow map within 5 to 10 minutes.

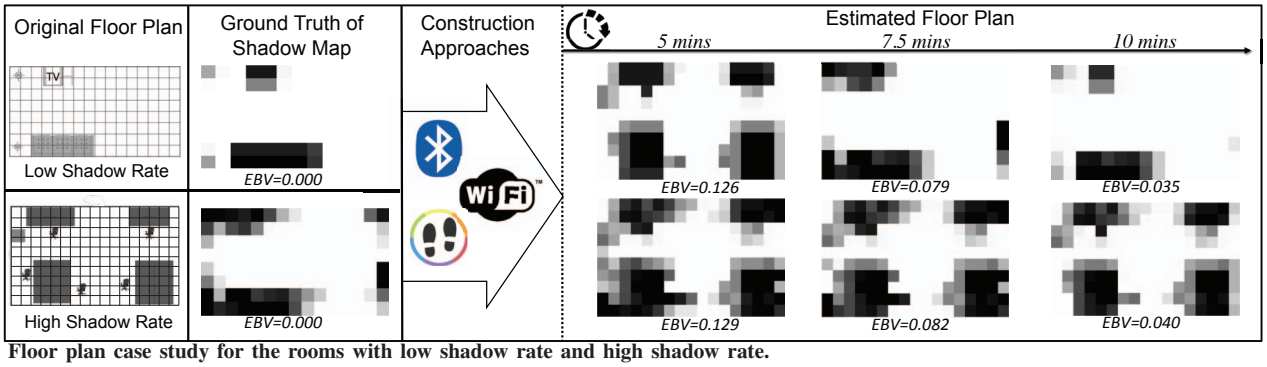


Fig. 10 Floor plan case study for the rooms with low shadow rate and high shadow rate.

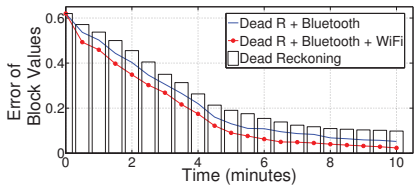
`iFrame`. Since no other person can establish a connection to him/her, only the dead reckoning technique is available; 2) Five participants use `iFrame`, combining the three approaches together. We repeat the experiments 12 times. Fig. 11(c) shows that with an increasing number of users, namely, more samples of matrices computed by `iFrame` can boost the accuracy of map construction. Additionally, the groups with more users cost less time to achieve a low error rate.

In our measurement, since the users of `iFrame` cover the rooms fully, the indoor maps are generated completely. However, in certain cases, the users may not pass by some areas in a room. These areas (named "blind area") cannot be described by our sensing approaches. To reduce the "blind area" on the map, once `iFrame` cannot receive the data samples from certain areas for 1 hour. `iFrame` sends a message to users to suggest to visit the "blind area".

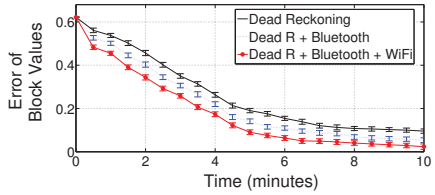
Different from letting volunteers walk frequently and cover the full space of eLANS lab within 10 minutes, we did an experiment in a living room of apartment (the size is known), which is closer to a real world scenario. Three users of `iFrame` enter and leave the room freely. Our system collected the data from three `iFrame` users in one day (from 10AM to 5PM), by applying the "blind area" messages, the shadow map can achieve a low Error of Block Value (0.041).

2) Building Measurement: In Figure 15, we extend our approach in a real building with multiple rooms and hallways. When `iFrame` constructs the floor plan of each single room, we also adopt the formula of combination mechanism to collect a, b, c values (Low/Normal/High shadow rate) for assigning weights to dead reckoning, Bluetooth and WiFi detections. The three groups of a, b, and c values are shown in Table II. Then, once a user enters a certain room, after first time period scanning, if the average shadow rate of a room is less than 0.1, it will choose a low shadow rate related a, b, c values. If the average shadow rate is more than 0.25, it will use a high shadow rate related values. Other cases will adopt normal shadow rate related values. As shown in Figure 12, although all of these experimental scenarios stay a low error level, the rooms with lower shadow rates have less errors.

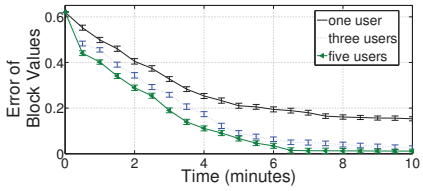
`iFrame` obtains improved results from combining the



(a) One time measurement by three approaches.



(b) Repeated measurements by three approaches.



(c) Control Group for Different Number of users.

Fig. 11 Experimental results for single room case study.

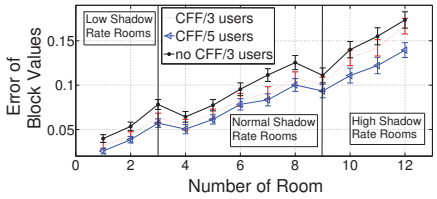


Fig. 12 Experimental results of the extended experiment.

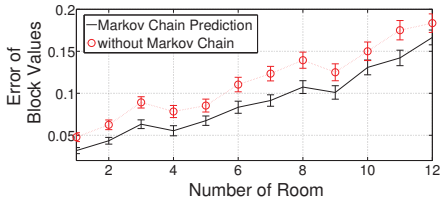


Fig. 13 Markov chain prediction comparison for multiple rooms.

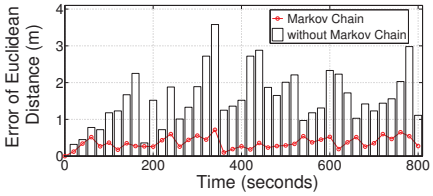


Fig. 14 Markov chain prediction comparison in one room.

TABLE II Different groups of a, b, and c values

	a	b	c
Low Shadow Rate	0.430	0.353	0.217
Normal Shadow Rate	0.412	0.324	0.264
High Shadow Rate	0.345	0.225	0.430

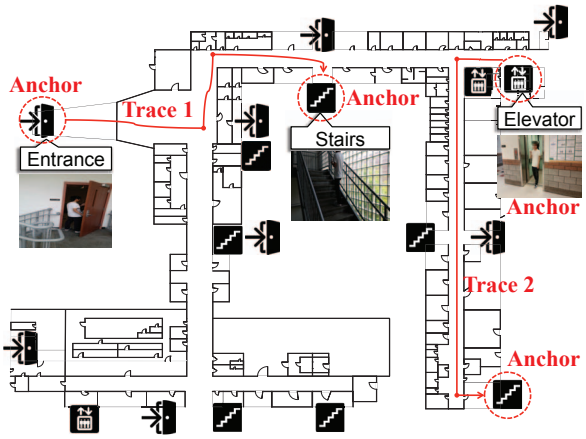


Fig. 15 Extend our evaluation from single room to the whole building via recognizing the anchor points.

three types of output matrices. The selection of a, b, c is based on CFF. To verify the effectiveness of CFF, we did the following comparisons: under the same condition as the previous evaluation, we set the values of a, b and c as 1/3, rather than selecting the values of a, b and c by the proposed method. Fig. 12 shows the EBV of 12 places without changing the initial values of a, b, c and the control group using CFF. We can make two interesting observations from the comparisons. First, the evaluation results confirm CFF enhances the accuracy of map construction effectively. Another observation is that the group with five users has better performance than the group having three users. Next, we study the benefit of Markov chain prediction, which aims to correct the deviations caused by dead reckoning. We still use the above experiment conditions, just deleting the Markov chain prediction mechanism and conducting `iFrame` 10 times in 12 rooms. As represented in Figure 13, we observe that the group of construction with Markov chain prediction outperforms the other one. Then, we only concentrate on the room 1 and run `iFrame` for 800 seconds. Two groups of results are illustrated in Figure 14. With the

TABLE III Confusion matrix for classifying different anchor positions

	Elevator	Stair	Entrance	Other	FP	FN	Total
Elevator	9	0	0	0	0%	0%	9
Stairs	0	20	0	1	0%	4.8%	21
Entrance	0	0	20	1	0%	4.8%	21
Other	0	0	0	40	5%	0%	40

time increase, the group with Markov Chain prediction has less error of deviation distance. Error of Euclidean Distance is the distance (in meters) between the ground truth and the estimated position.

The Anchor Points (e.g., entrances, stairs, and elevators) were identified in our system as shown in Fig. 15. Table III lists the recognition results for these Anchor Points. Only one stair and one entrance on our map were mis-classified by `iFrame`.

C. Discussion

In contrast to other map construction approaches, `iFrame` is a light-weight and low complexity application.

1) Running iFrame in the Cloud: Users of `iFrame` upload their accelerations and received RSSI values from cooperating with other devices periodically. The prepared distance-RSSI hash maps stored on the cloud server transfer the received RSSI values to distances continuously. To reduce computational burden on each smartphone, we deploy four critical tasks on the server instead of running it on smartphones: i) Forming the initial shadow map for a single smartphone, ii) Combine the matrices collected from various devices, iii) Refine the shadow map via Curve Fit Fusion (CFF), and iv) Filter the temporary shadow to enhance the accuracy of the indoor map.

2) Complexity: Although `iFrame` deployed tasks to a cloud, the cloud server does not need to process complex images or simulate people's micro-activities, such as shaking hands. In this subsection, we analyze the computational complexity of each corresponding task: i) `iFrame` adopts formula (2) and Markov chain prediction to generate the motion trace. Since the complexity of formula (2) is $O(n^2)$, the complexity of Markov chain prediction is $O(n^3)$ (the transfer matrix is $O(n^2)$, after the k steps' transitions, the complexity of Markov chain prediction is $O(k \times n^2)$). ii) We adopt the Mean Value model to merge the matrices built by different devices, the complexity of the Mean Value model is $O(n^2)$. iii) The complexity of the CFF is $O(n^2)$. iv) Conducting the crowd noise filter for each sub matrix, because the filter relies on k-means clustering, the

complexity of this approach is $O(n^{(di \times u+1)} \times \log n)$ (u is the number of cluster, di is dimension of each point to be clustered, n refers to the number of rooms to cluster). When we apply `iFrame`, the number of u, di are both set to three. Considering the number of rooms in `iFrame` is not beyond 500 for most of buildings, `iFrame` can manage to accurately construct the indoor map with such computational complexity.

IV. Related Work

SLAM: Many researchers have designed systems to rebuild indoor maps. SLAM (Simultaneous Localization And Mapping) [21]–[23] is a classical problem in the robotics area, which reconstruct maps in an unexplored environment. By using different sensors, such as gyroscopes, cameras, and laser sensors, the robots or other mobile devices can detect the layout of indoor environment. However, SLAM requires considerable computational power to sense a sizable area and process the complex collected data.

Smartphone Approaches: Since people use smartphones and other mobile devices in their daily lives, some approaches propose to build indoor maps by smartphones. In the computer vision area, researchers activate camera sensors on smartphones to construct the map [9], [24], [25]. JigSaw [9] leverages camera sensors to obtain images and adopts point cloud algorithms to piece together images to reconstruct the indoor floor plan. Sextant [24] combines the photos obtained by smartphones with gyroscope information to draw the map. Though these vision based approaches can build the indoor map by smartphones or other specific devices, the procedure of image processing is still challenging. Some work uses the dead reckoning approach by accessing data from accelerometers and gyroscopes to generate the trajectories of users [10], [11]. They analyze the trajectories and sensing data of elevators and entrances to build the map. Nevertheless, the deviations and errors of dead reckoning often reduce the accuracies. Other approaches use WiFi signatures and a series of algorithms to discover the layout of buildings, but the WiFi noise is difficult to process [26], [27].

Apart from SLAM or other smartphone-based approaches, `iFrame` does not require to process complex sensing and training data. `iFrame` users do not carry any extra devices except for smartphones. By combining RSSI analysis and dead reckoning of complementary strengths, `iFrame` constructs indoor maps and detects the changes of the indoor layouts effectively.

V. Conclusion and Future Work

We introduce a light-weight, unattended and high-speed indoor map construction approach called `iFrame`. After abstracting the unexplored map as a matrix, and by combining dead reckoning and RSSI detection techniques, `iFrame` judges whether the subareas in an indoor environment are empty or not. Each of proposed technologies compensates the shortcomings of others by adopting a matrix fusing mechanism. Our approach selects proper parameters for merging data automatically and yields a clear shadow map for each room within 5-10 minutes. By acceleration analysis and limited data training, we implemented `iFrame` in a real indoor building by running the applications on Android smartphones. `iFrame` reconstructs the layout of different rooms, hallways, and some special positions (as elevators, stairs, and entrances) successfully.

Although we achieved the goal of floor plan reconstruction on a two dimensional level, we plan to explore how `iFrame` may represent the heights of different objects to the future. Therefore, we may employ a three dimensional array to describe the environment. One dimension will record the height of the environment. Sensing approaches, such as Bluetooth, WiFi, and ultrasound may be used to measure both the positions and heights of smartphones. In addition, more types of Anchor Points, including seats in offices and hallways, will be recognized through body motion analysis.

Acknowledgment

The authors would like to thank Professor Dr. Max Mühlhäuser for his helpful comments. This work is supported in part by NSF Grant No. CNS-1320561.

References

[1] Gting, Ralf Hartmut, and Markus Schneider. Moving objects databases. Elsevier, 2005.

[2] Google indoor maps, available: https://support.google.com/gmm/answer/1685872?hl=en.

[3] Wang, He, et al. "No need to war-drive: unsupervised indoor localization." ACM MobiSys, 2012.

[4] Zheng, Yuanqing, et al. "Travi-navi: Self-deployable indoor navigation system." ACM MobiCom, 2014.

[5] Chatzimilioudis, Georgios, et al. "Crowdsourcing with smartphones." Internet Computing, IEEE 16.5 (2012): 36-44.

[6] Ni, Lionel M., et al. "LANDMARC: indoor location sensing using active RFID." Wireless networks 10.6 (2004): 701-710.

[7] Tung, Yu-chih, and Kang G. Shin. "EchoTag: Accurate InfrastructureFree Indoor Location Tagging with Smartphones." ACM MobiCom, 2015.

[8] Yang, Zhice, et al. "Wearables can afford: Light-weight indoor positioning with visible light." ACM MobiSys, 2015.

[9] Gao, Ruipeng, et al. "Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing." ACM MobiCom, 2014.

[10] Alzantot, Moustafa, and Moustafa Youssef. "CrowdInside: automatic construction of indoor floor plans." ACM SIGSPATIAL, 2012.

[11] Philipp, Damian, et al. "MapGENIE: Grammar-enhanced indoor map construction from crowd-sourced data." IEEE PerCom, 2014.

[12] Steinhoff, Ulrich, and Bernt Schiele. "Dead reckoning from the pocket-an experimental study." IEEE PerCom, 2010.

[13] Hossain, AKM Mahtab, and Wee-Seng Soh. "A comprehensive study of bluetooth signal parameters for localization." IEEE PIMRC, 2007.

[14] Chintalapudi, Krishna, et al. "Indoor localization without the pain." ACM MobiCom, 2010.

[15] CH Robotics LLC, UM6 Orientation Sensor, available: http://www.chrobotics.com/library/accel-position-velocity.

[16] Gardiner, Crispin W. Handbook of stochastic methods. Vol. 4. Berlin: Springer, 1985.

[17] Weppner, Jens, and Paul Lukowicz. "Bluetooth based collaborative crowd density estimation with mobile phones." IEEE PerCom, 2013.

[18] Xi, Wei, et al. "Electronic frog eye: Counting crowd using wifi." IEEE INFOCOM, 2014.

[19] Noom Walk Pedometer, available: https://www.noom.com/walk.php.

[20] S Health Pedometer, available: https://shealth.samsung.com.

[21] Brand, Christoph, et al. "Stereo-vision based obstacle mapping for indoor/outdoor SLAM." IEEE IROS, 2014.

[22] Luo, Ren C., and Chun C. Lai. "Enriched indoor map construction based on multisensor fusion approach for intelligent service robot." IEEE Transactions on Industrial Electronics, 59.8 (2012): 3135-3145.

[23] Google Tango, available: https://www.google.com/atap/project-tango/.

[24] Tian, Yang, et al. "Towards Ubiquitous Indoor Localization Service Leveraging Environmental Physical Features." IEEE INFOCOM, 2014.

[25] Sattler, Torsten, Bastian Leibe, and Leif Kobbelt. "Fast image-based localization using direct 2D-to-3D matching." IEEE ICCV, 2011.

[26] Jiang, Yifei, et al. "Hallway based automatic indoor floor plan construction using room fingerprints." ACM UbiComp, 2013.

[27] Shen, Guobin, et al. "Walkie-markie: indoor pathway mapping made easy." USENIX NSDI, 2013.