SenseWit: Pervasive Floorplan Generation Based On Only Inertial Sensing

Jiaqi Liang, Yuan He, Yunhao Liu
School of Software and TNLIST, Tsinghua University

***Abstract*— Mobile crowdsourcing is deemed as a powerful technique to solve traditional problems. But the crowdsourced data from smartphones are generally with low quality, which induce crucial challenges and hurt the applicability of crowdsourcing applications. This paper presents our study to address such challenges in a concrete application, namely floorplan generation. In order to utilize pedestrians' traces for indoor location inference, existing proposals mostly rely on infrastructural references or accurate data sources, which are by nature restricted in terms of applicability and pervasiveness. Our proposal called SenseWit is motivated by the observation that people's behavior offers meaningful clues for location inference. The noise, ambiguity, and behavior diversity contained in the crowdsourced data, however, mean non-trivial challenges in generating high-quality floorplans. We propose 1) a novel concept called Nail to identify featured locations in indoor space and 2) a heuristic pathlet bundling algorithm to progressively discover the internal layouts. We implement SenseWit and conduct real-world experiments in different spaces. Our work offers an efficient technique to obtain high-quality structures (either logical or physical) from low-quality data. We believe it can be generalized to other crowdsourcing applications.**

I. Introduction

Crowdsourcing refers to the process of obtaining needed service by soliciting contributions from a large group of volunteers. It offers a new way to accomplish many jobs that are previously considered to be complex or cost-intensive in traditional fields.

As a popular item in people's daily life, smartphones are naturally involved in people's daily activities and living/working spaces. Utilizing smartphones for crowdsourcing, i.e. mobile crowdsourcing, becomes a promising direction [1]. There are many proposals for research and application of mobile crowdsourcing, such as localization [2][3], map generation [4], floorplan generation [5][6], etc..

Most smartphones have equipped with inertial sensors (*e.g.*, accelerometer, gyroscope, compass). These sensors can finish the same task while consuming less energy and user concern. But they face the challenges of low data quality. Generally, the data quality may be interpreted in three levels: 1) accuracy and precision of measurements; 2) quantity and density of data, with respect to the application requirement; 3) fidelity and consistency of results, contrasted to the ground truth.

This paper presents our study to address the above challenges in a concrete application called floorplan generation. A floorplan is a diagram of the arrangement of rooms and layouts in one floor of a building. With the popularity of location-based services, floorplans are in great demand. But they are not available in many contexts, due to commercial or security-related reasons. Floorplan generation via mobile crowdsourcing thus becomes a desirable technique.

Floorplan generation is partially similar with the problem of map generation [4]. Given the ability of localization, the common idea behind map generation is to connect discrete points into curves, which then forms an outdoor map. Since indoor localization is a non-trivial issue, existing proposals of floorplan generation mostly rely on infrastructural references or accurate data sources. For example, [7] collects measurements of signal strength to a couple of WiFi access points. [8] requires to mount inertial sensors on user bodies for precise motion sensing. Note that for people in a venue where the floorplan is unknown, it is very likely that the above-mentioned data sources are unavailable. How to generate floorplans under such scenarios remains an open problem.

Our idea is motivated by the observation that people's behavior offers meaningful clues for location inference. From a statistical point of view, people make turns at corners and stay stationary for a short in particular positions like water dispenser. Locations corresponding to those behavioral features may be labeled, called *featured locations*. Using only inertial sensing to identify people's behavior, one can crowdsource plenty of people's movement pathlets with *featured locations* labeled on them. Intuitively, if one bundles those labeled pathlets together, a complete floorplan can be generated progressively.

Towards this goal, we need to address several critical challenges: First, the noise in inertial sensing, caused by either hardware diversity or motion instability, often blurs the features of behavior data; Second, diversity of people's behavior often causes false identifications of featured locations; Third, a featured location, even correctly identified, sometimes may correspond to multiple locations in the space, as is called *ambiguity* of labels.

Our work in this paper—SenseWit, is an efficient technique to generate indoor floorplans while coping with the above challenges. Our contributions can be summarized as follows.

(1) We present both opportunities and challenges in utilizing inertial sensing data. We propose *Nail* to identify featured locations in indoor space based on the raw sensor data.

(2) We design an efficient pathlet bundling algorithm, called TriNail matching, to generate floorplan with pathlets. The proposed algorithm is robust to noise, ambiguity, and diversity of people's behavior. We tackle a typical problem—*analogous sub-structures*, which is common in indoor environment.

2325-2944/16 $31.00 © 2016 IEEE
DOI 10.1109/DCOSS.2016.25

(3) We implement SenseWit and evaluate it through real-world experiments. The result demonstrates that SenseWit generates accurate floorplans even with limited data, while the cost of computation, storage, and energy is relatively low.

The rest of this paper is organized as follows. In Section II we discuss the related work. Section III elaborates on the system design and detailed solutions. Section IV shows the implementation and evaluation results. In Section V, we show our future work and conclude this paper.

II. Related Work

A. Mobile Crowdsourcing

Mobile crowdsourcing has been used in many traditional fields, such as traffic management [9], localization [3][10], etc.. Shu *et al.* [11] try to solve the last-mile navigation problem using crowdsourced leader's data to guide followers, and they point at the quality control problem. CrowdMap [12] provides a crowdsourcing system to construct indoor floorplan. But it relies on the sensor-rich videos uploaded by the crowd, improving accuracy at the cost of energy consumption and network traffic.

B. Inertial and Motion Sensing

Inertial sensors consume less energy than traditional sensors such as camera, WiFi and GPS, likely to be used in localization and navigation [10][13]. The most significant challenge in inertial sensing is the accumulation of errors [14] and error caused by phone's attitude [15]. Zee [16] addresses this problem by leveraging the constraints imposed by maps to filter out erroneous measurements. However, it requires the indoor floorplan as a known condition.

Some studies focus on how to get rid of error. For example, WalkCompass [17] proves that peak recognition can be used to detect strikes/steps more accurately. In FollowMe [11], they combine accelerometer and gyroscope, using the basic idea that rotation axis of the body during a turn is always directed toward the center of earth to detect turn degree.

C. Map Generation and Floorplan Generation

Map generation and floorplan generation are two similar problems. For map generation, there have been many studies. SLAM [18] builds a map while tracking the current location, but the map constructed only consists of significant points. SmartSLAM [8] applies this technique on smartphones and utilizes the images collected when users are walking. Floorplan generation is more challenging, since it is more difficult to obtain accurate coordinates indoors. Jiang *et al.* [19] propose an automatic floorplan generation method based on hallway, but it mainly depends on WiFi fingerprints to function. Jigsaw [6] proposes a system which combines traces and location excavated from images to reconstruct floorplan. But it requires high labor cost to obtain the images and has a high demand on image quality. Moreover, the use of camera makes it energy-intensive. CrowdInside [5] adopts the dead-reckoning technique and enhances the accuracy by using unique anchor points. However, it makes high demand on traces and requires a uniform starting point like an entrance. Differing from the state of arts, SenseWit does not rely on either the infrastructural references or accurate data sources. It is able to generate high-quality floorplans even in face of the noisy data from mobile crowdsourcing.

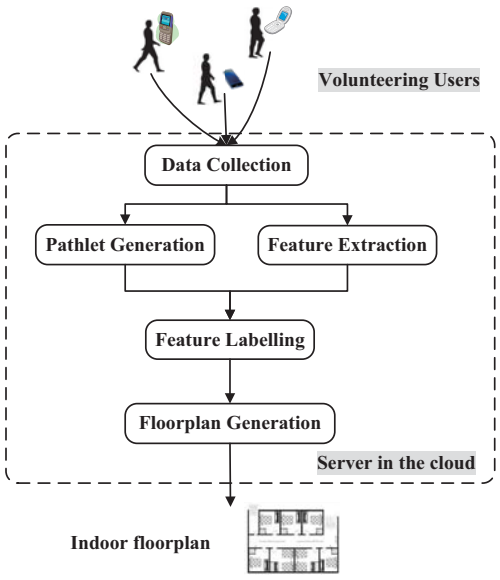


Fig. 1: The architecture of SenseWit.

III. Design

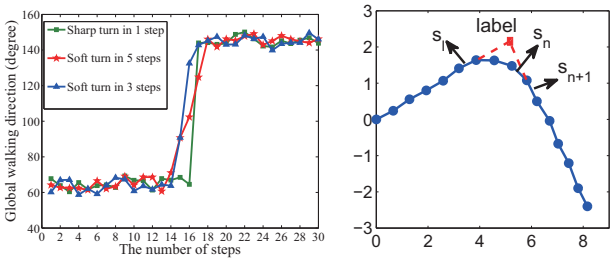
Fig. 1 describes the architecture of SenseWit. The work flow involves volunteering users and a floorplan generation server in the cloud. Data are collected from inertial sensors and uploaded when network is available. On the server, motion pathlets are generated, features are extracted, and then the featured locations (Nails) are labeled. Based on those Nails, pathlets are bundled together according to pathlet bundling algorithm and a complete floorplan is progressively generated. We introduce details of the above process in this section. Meanwhile, we have analysis concerning the efficacy and efficiency.

A. Feature Recognition and Labeling

1) Data Collection:

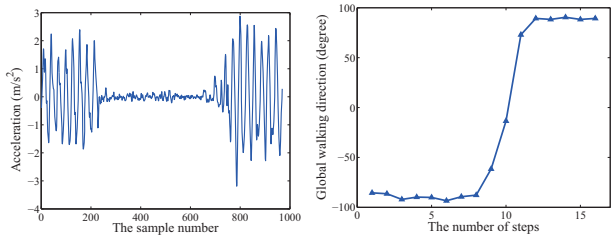
In SenseWit, we use three inertial sensors—accelerometer, gyroscope, and compass. Accelerometer and gyroscope are used to compute steps and generate movement pathlets. Compass can fix the pathlets into a certain direction. Note that we employ periodical checking instead of collecting data all the time. The collection process is triggered when a step is detected, and is stopped when people are static for a certain period or the collection time exceeds a threshold.

The raw data are processed first to reduce noise. Knowing that the frequency of most human activities is below 15Hz [20], we first leverage a low-pass filter with a 15Hz cut-off to remove the high-frequency component of data noise. Second, we calculate the sum vector of both acceleration and angular



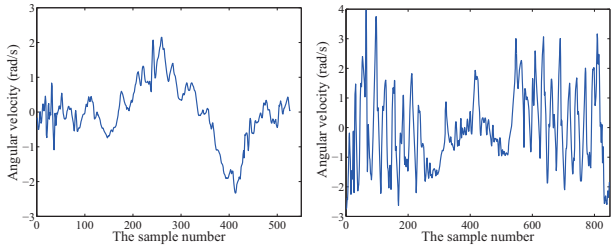
(a) Walking direction of turning (b) Turning labeling method

Fig. 2: Feature recognition and labeling of turning.



(a) Acceleration (b) Walking direction

Fig. 3: Features of water dispenser.



(a) Palm/phoning/texting (b) Swinging hand

Fig. 4: Features of door in different phone placements.

velocity. Last but not least, since the reading of accelerometer is influenced by gravity, a non-zero DC component is removed.

2) Feature Recognition:

We first use a sliding window of 128 samples (2.56 seconds in 50Hz) to segment the preprocessed data. The effectiveness has been shown in the previous work [21]. For each segment, a decision tree is used to identify motion state (walking, keeping stationary or irregular). Continuous segments with the same state are joined together. When the state is walking, step is detected and walking direction is estimated. Otherwise, we recognize a water dispenser and a door according to the corresponding feature specifications, as are introduced in the subsequent content. It is worth mentioning that we use the above three features as an example to illustrate our idea, but our approach is not limited to using any specific feature. Different features (elevator, the reception desk, etc.) can also be used in different indoor spaces.

Turning: The most direct way to recognize a turning is to calculate the change in walking direction. However, both the diversity of people's behavior and the error introduced by direction estimation cause interference.

The biggest difference reflected on people's behavior is "soft turn" and "sharp turn". As shown in Fig. 2(a), people walk along the same way while taking 1 step, 3 steps, and 5 steps respectively to pass the turning point. When people make a soft turn, each amplitude of variation is smaller than a sharp turn. So we take two ways to recognize a turning:

First, a sharp turn is recognized if $|d_{s_{n+1}} - d_{s_n}| \geq d_\alpha$, where s_n and s_{n+1} are two sequential steps and d is the walking direction. Second, if we detect $|d_{s_{n+1}} - d_{s_n}| \geq d_\beta$, we record and analyze the following directions. If the direction change exceeds d_α in 6 steps, we consider it as a soft turn. Otherwise, we ignore it. We choose 6 steps because people usually make a turning in less than 6 steps [17]. We use $d_\alpha = 75°$ and $d_\beta = 20°$, which yield good results in our experiments.

Water Dispenser: When people walk to a water dispenser or a reception, a relatively long stationary period and a direction change can be extracted as feature, as shown in Fig. 3. The recognition process starts when the state transits from walking to the stationary state. In order to distinguish this feature from other behavioral interference, such as a temporary stop or sitting, we propose two measures. First, If the stationary duration is longer than a threshold t_α, it may represent sitting and is not what we need. Otherwise, we get the walking direction and judge if the change is larger than d_γ. Only when these two conditions are satisfied, we recognize this location effective. According to our empirical observation, we set $t_\alpha = 20s$ and $d_\gamma = 165°$ in the experiment.

Door: When people are walking through a door, they tend to slow down, open the door, then make a turning and close the door. These continuous motions can be recognized as feature. However, interfered by the placement and orientation of the phone, the features are probably blurred.

According to experiments, when people are holding the phone in the palm, texting or phoning, the angular velocity always presents similar characteristics, including a pair of prominent crest and valley, as shown in Fig. 4(a). But when people are holding the phone in a swinging hand, the features are quite different, as shown in Fig. 4(b), where the two slow valleys are distinguished from the normal periodical motions. We introduce our scheme to cope with phone placement in the following part, so that these two situations can be distinguished and the features of door can be recognized.

Coping with ambiguity: Ambiguity caused by different poses of phone placements is useful for feature recognition. Here we borrow the classifier in [21] to judge the type of phone placements. Once placement is known, we estimate the walking direction and use it to recognize other features.

3) Feature Labeling:

Given a walking pathlet shown in Equation. 1, where n is the total steps, how to label Nails on it?

$$P = \{(0,0),(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}, \quad (1)$$

For water dispenser, door, and sharp turn, we label the starting point of the last step. For a soft turn, we take a different method. As shown in Fig. 2(b), d_{s_l} is the direction of step s_l, which is the last step before turning. And d_{s_n} is the step that reveals the existence of turning. Two lines can be drawn according to d_{s_l} and $d_{s_{n+1}}$ and the intersection of lines is calculated, marked with $label$. $label$ is considered as the location of this turning. We choose step s_{n+1} instead of s_n because people might not turn completely in s_n.

The Nail set corresponding to each pathlet is described as:

$$\{\{l_1,(x_1,y_1)\},\{l_2,(x_2,y_2)\},...,\{l_k,(x_k,y_k)\}\}, \quad (2)$$

where k is the number of features on this pathlet and l_k represents the label of feature.

B. Floorplan Generation

In this section, we first introduce details about *TriNail*, including the definition, the reason we choose it, and the pathlet matching strategy. Then we demonstrate how to evaluate the priority of *TriNail* classes and the pathlet bundling algorithm. Finally, the floorplan shaping process is presented.

1) TriNail Overview:

Definition of TriNail: A TriNail is defined as a virtual triangle formed by three non-collinear *Nails* on a certain pathlet, which can be described with a feature vector:

$$TriNail(i,P_k)=(l_1,l_2,l_3,e_{12},e_{13},e_{23}),1\le k\le m \quad (3)$$

P_k is the kth pathlet and i denotes the ith TriNail on P_k. l are the labels of *Nails*. e represent the features of an edge, including the edge's length and orientation.

Reason for Choosing TriNail: The main role of TriNail is to improve the pathlet bundling accuracy. Influenced by feature recognition and labeling error, not all the pathlets carry correct information. If we use incorrect pathlets as references to bundle others, there will be paradoxes in the result. Note that three points are the minimum to achieve a stable planar structure. Using three Nails for matching enables one to learn stable structures as efficient as possible, while eliminates the interferences of paradoxes. We use TriNail in pathlet priority assessment, ensuring the reliability of bundled pathlets. However, TriNails still meet a problem called *analogous sub-structures*, meaning that though pathlets own the same feature vector, they belong to different indoor regions and should not be bundled. We propose an improved method to resolve this problem in Sec. III-B2.

TriNail Matching Principle: In principle, two TriNails are *matched* only if they have the same feature vectors. But it is hard to find two TriNails that have exactly same feature vectors due to the localization error. Therefore, we device an approximation strategy to tolerate errors.

For two TriNails A and B, they are considered to be matched and belong to the same TriNail class if and only if:

$$\frac{S(A\cap B)}{\max\{S(A),S(B)\}}\ge\alpha \; and \; f(A)=f(B) \quad (4)$$

Algorithm 1 Pathlet Bundling Algorithm

Input: m useful pathlets $\{P_k\},1\le k\le m$
and pathlets $\{p\}$ that do not contain TriNails
Output: An indoor floorplan F

```
Construct TriNail set S{P_k};
Classify S{P_k} into classes S{C} = {C_1, C_2, ..., C_n};
Select a seed C_s ∈ S{C};
repeat
    Bundle pathlets {P_a, P_b, ...} ∈ P_k based on C_s;
    Delete C_s from S{C};
    Refresh the Matrix;
    if ∃{C_i} ∈ S{C} appears in both {P_a, P_b, ...}
    and {P_k} − {P_a, P_b, ...} then
        select C_s ∈ {C_i} that appears most in {P_k} −
        {P_a, P_b, ...} as the seed;
    else
        select C_s ∈ S{C} as the seed;
    end if
    Delete {P_a, P_b, ...} from {P_k};
until S{C} = ∅ or {P_k} = ∅
if more than one set exist
    and the sets can be connected through {p} then
    bundle the sets;
end if
```

where $S(A)$ and $S(B)$ represent the area of A and B. $S(A\cap B)$ is the maximum common area. α is a threshold that constrains the similarity of TriNail's shape, which is obtained through empirical experiments. If the ratio exceeds α, we define them as similar in shape. $f(A)=f(B)$ means that the matched TriNails have completely same labels.

2) Pathlet Bundling Algorithm:

Priority of TriNail Class: TriNails are converted into discrete classes through the matching principles, and a TriNail class includes all the matched TriNails having the same labels and similar shape. The relationship between classes and pathlets are acquired, so that the occurrence frequency of each class can be described by a matrix:

$$M=\begin{array}{c} \\ C_1\\ C_2\\ \vdots\\ C_k\end{array}\begin{array}{c}\begin{array}{cccccc}P_1 & P_2 & \dots & P_m & Total\end{array}\\ \begin{pmatrix}1 & 0 & \dots & 2 & N_1\\ 0 & 1 & \dots & 0 & N_2\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 1 & 1 & \dots & 0 & N_k\end{pmatrix}\end{array}$$

The elements in the matrix represent how many times the TriNail class $C_i(1\le i\le k)$ appears on pathlet $P_j(1\le j\le m)$, and the last column represents the total number of times this class appears.

There are still two challenges in pathlet bundling. First, errors occur in feature recognition, leading to wrong labels. A TriNail with label $\{1,2,1\}$ might be judged as $\{1,2,2\}$, or an extra label is added although it does not exist at all. Second, TriNail matching does not eliminate all the ambiguity, due to the issue induced by *analogous sub-structures*.

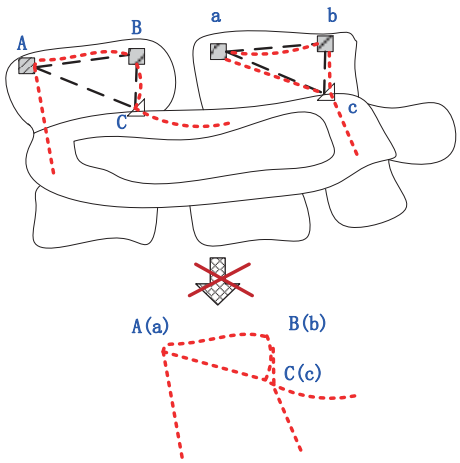
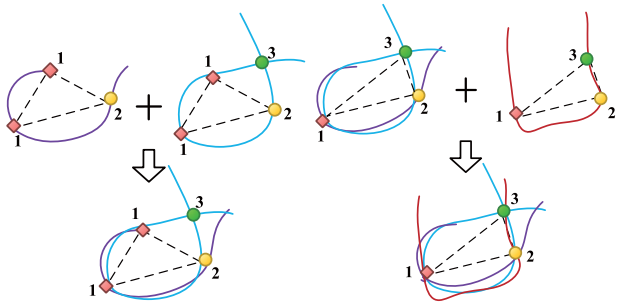


Fig. 5: Analogous sub-structures problem.



(a) First round (b) Second round

Fig. 6: Two rounds of pathlet bundling.

Analogous sub-structures problem is quite common in indoor space. Figure. 5 presents a general floorplan of a shopping mall. There exist two pathlets: one has three Nails $\{A, B, C\}$, another with three same Nails $\{a, b, c\}$, and the path segments are all identical. But these two pathlets cannot be bundled because they are actually in different regions. If they are bundled, the result will be wrong.

In order to solve the above problems, SenseWit runs in an iterative manner. In each round, we select the TriNail class with the highest priority and bundle pathlets based on it. There are three principles to determine the priority of TriNail class:

(1) The class that appears more frequently on the remaining pathlets is given higher priority. This principle is designed to deal with the label error, for the occurrence of a wrong label is actually a special case. If two classes have same times of occurrences, the one with a larger area is prioritized.

(2) If the occurrence frequency of one class is less than a threshold, we discard the corresponding pathlets, for we think these pathlets are not reliable.

(3) If matrix element (except the total number) in the row C_i is larger than 1, there are two possibilities. First is people walk the same way more than once. Second, it is an indication of *analogous sub-structures*. Therefore, we consider this class as untrustworthy and also discard the corresponding pathlets.

Intuitively, the lower priority a TriNail class has, the later it is used for pathlet bundling. Therefore, the negative effect of error is limited as much as possible.

Algorithm Flow: The algorithm works in a greedy manner, as described in Algorithm. 1. In each round, a TriNail class with the highest priority is selected as the footstone, called a "seed", and all the pathlets containing the seed are bundled.

Fig. 6 illustrates two rounds of our algorithm. The TriNails are denoted using dashed line while the labels are represented with different icons and numbers. In the first round, the pathlets are bundled based on the TriNail with label sets $\{1, 1, 2\}$. Next, another TriNail with label sets $\{1, 3, 2\}$ can be found on the bundled pathlets. Based on this TriNail, the third pathlet is bundled together in the second round.

It is possible that we cannot find a seed connecting the bundled pathlets and the unbundled ones. In this situation, we select a new seed from all the seeds and the bundling process continues. The process goes on until no more edges or points can be added. If there exists more than one set in the final stage, we look for the unused pathlets to check if a "bridge" can connect any two disjoint sets. As long as the indoor space is connected and the crowdsourced data cover the entire area, SenseWit is able to generate a complete floorplan.

3) Floorplan Shaping:

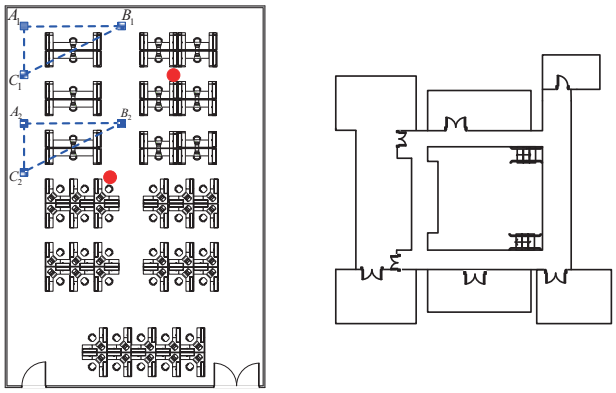
After the above steps, chaotic pathlets have been processed and bundled together. In this way, a rough floorplan is generated. To provide a better visualized result, we go one step further and utilize the technique of occupancy grid map [6][22] to construct the hallway and room structure.

C. Analysis on the Overhead

Computational Complexity: We analyze the worst case when only two pathlets are bundled in each round. Suppose the total number of Nails on all the pathlets is N, there are at most $\binom{N}{3}$ TriNails, which is $\frac{N^3-3N^2+2N}{6}$. Therefore the time complexity is bounded by $O(N^3)$. It is worth noticing that the practical convergence speed is much faster than that the worst case, because every popular TriNail can bundle a large number of pathlets in one round.

Storage Cost: Limited by our method of periodically checking and triggering, the size of each record is usually hundreds of kB. And the processed pathlets are usually less than $10kB$. Common servers are sufficient to store at least millions of sensor readings and pathlets, which is much more than enough to generate a complete floorplan.

Energy and Privacy: We use inertial sensors instead of power-wasting ones (camera, radio, GPS), and employ periodical checking and conditional triggering instead of always on. Therefore, the energy cost can be reduced to an acceptable degree. Moreover, we do not need users' personal information to analyze the data, avoiding leakage of user privacy.

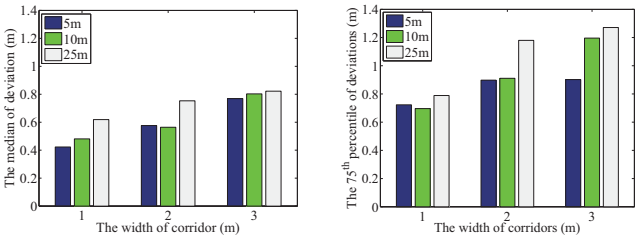


(a) Scenario 1 (b) Scenario 2

Fig. 7: The ground truth of two scenarios.

TABLE I: Confusion matrix of feature recognition.

	Turning	Door	Water	Null	Recall
Turning	**73**	0	0	3	96.1%
Door	5	**40**	0	0	88.9%
Water	0	0	**23**	4	85.2%
Precision	93.6%	100%	100%	-	-



(a) The median (b) The 75^{th} percentile

Fig. 9: Influence of hallways' width and length.

IV. Evaluation

We first introduce our experimental environments, followed by evaluation on the feature recognition accuracy, pathlet labeling accuracy, and floorplan generation performance.

A. Experimental Environments

We implement SenseWit on different Android phones (Samsung Galaxy S5, HTC ONE M8, Millet 3) and conduct experiments in two scenarios: an office of $24m \times 19.2m$ with 2 doors, 2 water dispensers (marked with red dots) and more than 10 turnings; and one floor in a campus library with $464m^2$ area, having 6 rooms. Fig. 7 is the ground truth. 10 volunteers are invited in two scenarios respectively. The sampling rate is SENSOR_DELAY_GAME (50Hz).

B. Feature Recognition Accuracy

We select 30 typical pathlets with 148 featured locations to evaluate the accuracy of feature recognition. The result is shown in Table I. The row denotes the real features, while the column is the recognition results. The diagonal indicates the number of correctly classified features, while "Null" column indicates failing to recognize. We can see that the recalls are all higher than 85% and the precisions are more than 90%. Some doors are wrongly recognized as turning because the door opening/closing motions are often along with turning. And the missing water dispenser might be caused by particular behavior (such as waiting for long time when fetching water).

C. Pathlet Labeling Accuracy

Euclid distance is used to compute the location deviations and Fig. 8 shows the cumulative distributed function for turning, water dispenser, and door. We can see that though a small proportion of large deviations ($1.8m$) occur for water dispenser and door, around 90% of the results have deviations less than $1m$. For turning, 70% of the deviation is under 1m and 90% less than $1.5m$, demonstrating a good accuracy. There are a spot of deviations larger than $2m$ in turning, caused by people who sometimes walk along the border of corridor, but this only accounts for less than 3% of all measurements.

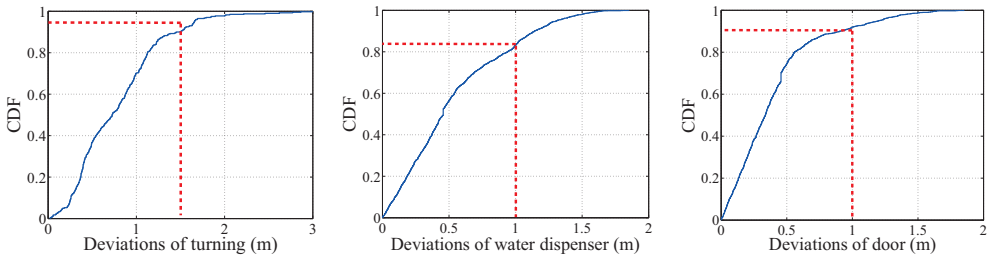
We present more details to measure the influence of different users and placements. Due to the page limit, we only take turning as an example to illustrate the results.

First we measure the influence of hallways' width and pathlet length. Fig. 9 shows the median and the 75^{th} percentile for 9 situations. The three bars in each width represent the walking length before turning. We can see that although the location deviation grows a bit larger when hallway becomes wider, it keeps in a low error level of around $0.8m$ in median and $1.2m$ for 75^{th} percentile. Considering that the hallways are commonly less than $3m$ in daily life, our method is robust to different environments.

Fig. 10 presents the deviations of 5 users, illustrating the robustness to different phone poses. We select a $2m$ hallway and the users walk for $10m$ before turning. When people hold the phone in hand or make a phone call, the median and 75^{th} percentile are around $0.5m$ and $0.8m$. In the case of swinging hand, the deviations are a bit higher, i.e. $0.6m$ and $0.9m$. This is because the direction error of swinging is larger than the other two poses. In addition, the location deviation of user 4 is particularly higher than others, caused by his way of walking (he tends to walk along the margin of the hallway). Here we only consider three typical phone poses, and more poses (*e.g.*, in the pocket) are our future work.

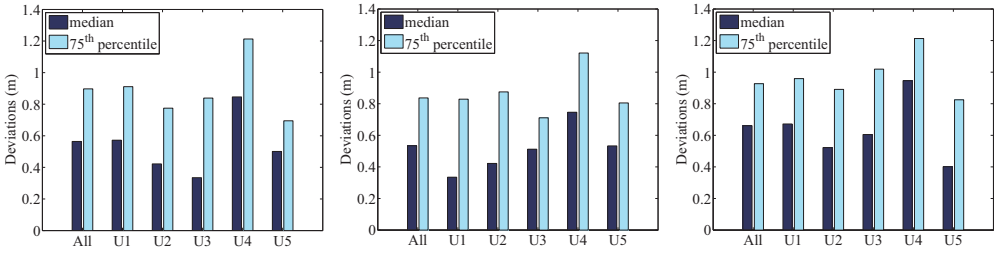
D. Floorplan Generation Performance

Fig. 11 presents the process of pathlet bundling and the output of floorplan in the first scenario. In Fig. 11(a), the first "seed" is a TriNail formed by a water dispenser, a door and a turning. Among the bundled pathlets, there exists another TriNail, which is selected as the second "seed", marked in Fig. 11(b). The procedure goes on and the final result is shown in



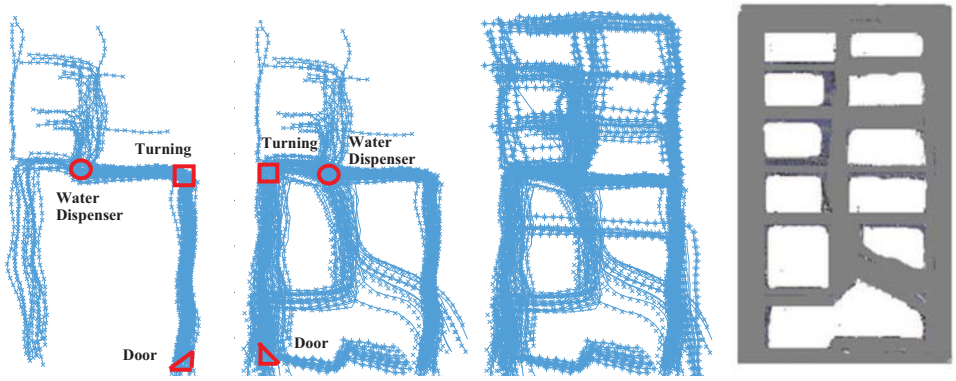
(a) Turning (b) Water dispenser (c) Door

Fig. 8: Labeling deviations.



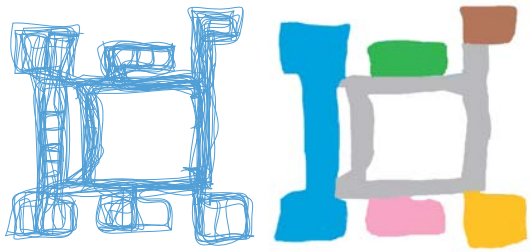
(a) Phone held in hand (b) Making a phone call (c) Swinging hand

Fig. 10: The median and 75^{th} percentile of location deviations in turning.



(a) First Round (b) Second Round (c) Pathlet Bundling Result (d) Shaped Floorplan

Fig. 11: Floorplan generation of the first scenario.



(a) Pathlet bundling result (b) Shaped floorplan

Fig. 12: Floorplan generation of the second scenario.

Fig. 11(c), where the dotted lines together draw the hallways. The blank area represent stationary objects like tables, seats, or other obstacles. The final floorplan is presented in Fig. 11(d). Fig. 12 shows the generated floorplan of the second scenario. There are 6 detached regions in this space, and 5 of them are recognized. Two detached rooms in the left are combined as one because the door is always open, resulting that people behave the same as elsewhere in the hallway.

We evaluate the performance from two aspects: hallway shape and room size, and compare SenseWit with CrowdInside [5], Jigsaw and CrowdInside++ [6].

Hallway Shape: We adopt the same metric as Jigsaw [6] to evaluate the hallway shape similarity, and the result is

TABLE II: Evaluation of hallway shape.

	Scenario.1	Scenario.2	CrowdInside++	Jigsaw
Precision	78.5%	72.1%	64.0%	77.8%
Recall	84.5%	80.3%	48.2%	90.4%
$\mathcal{F}$	81.4%	76.0%	54.9%	83.6%

TABLE III: Evaluation of room size.

	SenseWit	Jigsaw	CrowdInside
Error	31.4%	27.6%	42.7%

shown in Table II. Precision is the ratio of the overlapping area to the generated hallway. Recall stands for the ratio of the overlapping area to the ground truth. $\mathcal{F}$ represents the harmonic mean of the precision and recall. We can see that SenseWit is much better than CrowdInside++, while a little poorer than Jigsaw. But Jigsaw consumes more energy and achieves its accuracy at high labor cost.

Room Size: We use room area error, which is defined as the area difference between the generated room layout and the ground truth divided by the ground truth, to evaluate our method. The result is shown in Table III. SenseWit is better than CrowdInside and comparable to Jigsaw by consuming less energy and labor cost. The error is mainly caused by obstacles, where people's movements can not cover.

V. Conclusion and Future Work

Smartphones get proliferated nowadays. How to utilize the sensing capability of smartphones becomes an increasingly important issue. This paper presents our research effort to employ crowdsourcing technique in the application scenario of floorplan generation. Based on the insightful finding that people's behavior can be used as meaningful clues for location inference, we propose a mobile crowdsourcing based approach to efficiently generate floorplans. The design and implementation of SenseWit involve successful practice to address practical challenges in utilizing crowdsensed data, such as noise, ambiguity, and diversity of people's behavior. We believe this work acts as an example of using crowdsourcing to solve traditional hard problems. The methodology of obtaining stable structures (either logical or physical) from unstable data may be generalized to many other application scenarios.

We leave some further study in the future. First, we only use specific types of behavior in this work. More representative features of people's behavior can be exploited to tailor our approach for more scenarios. Second, we plan to carry out more experiments in various spaces, e.g. shopping malls and restaurants, and build a publicly available smartphone application for ordinary users.

Acknowledgment

This research is supported in part by the National Natural Science Fund for Excellent Young Scientist under grant No. 61422207 and the National Natural Science Foundation of China under Grant No. 61373146.

References

[1] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *Acm Computing Surveys*, vol. 48, no. 1, 2015.

[2] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *ACM MobiCom*, 2010.

[3] W. Sun, J. Liu, C. Wu, Z. Yang, X. Zhang, and Y. Liu, "Moloc: On distinguishing fingerprint twins," in *IEEE ICDCS*, 2013.

[4] X. Chen, X. Wu, X. Li, Y. He, and Y. Liu, "Privacy-preserving high-quality map generation with participatory sensing," in *IEEE INFOCOM*, 2014.

[5] A. Moustafa and Y. Moustafa, "Crowdinside: Automatic construction of indoor floorplans," in *ACM SIGSPATIAL*, 2012.

[6] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li, "Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing," in *ACM MobiCom*, 2014.

[7] M. Azizyan, I. Constandache, and R. R. Choudhury, "Surroundsense: Mobile phone localization via ambience fingerprinting," in *ACM MobiCom*, 2009.

[8] H. Shin, Y. Chon, and H. Cha, "Unsupervised construction of an indoor floor plan using a smartphone," *IEEE Transactions on Systems, Man, and Cybernetics Society*, vol. 42, no. 6, pp. 889–898, 2012.

[9] P. Zhou, Y. Zheng, and M. Li, "How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing," in *ACM MobiSys*, 2012.

[10] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *ACM MobiSys*, 2012.

[11] Y. Shu, K. G. Shin, T. He, and J. Chen, "Last-mile navigation using smartphones," in *ACM MobiCom*, 2015.

[12] S. Chen, M. Li, K. Ren, and C. Qiao, "Crowd map: Accurate reconstruction of indoor floor plans from crowdsourced sensor-rich videos," in *IEEE ICDCS*, 2015.

[13] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-markie: Indoor pathway mapping made easy," in *USENIX NSDI*, 2013.

[14] S. Beauregard and H. Haas, "Pedestrian dead reckoning: A basis for personal positioning," in *IEEE WPNC*, 2014.

[15] L. Zhang, K. Liu, Y. Jiang, X. Li, Y. Liu, and P. Yang, "Montage: Combine frames with movement continuity for realtime multi-user tracking," in *IEEE INFOCOM*, 2014.

[16] R. Anshul, K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *ACM MobiCom*, 2012.

[17] N. Roy, H. Wang, and R. R. Choudhury, "I am a smartphone and i can tell my users walking direction," in *ACM MobiSys*, 2014.

[18] H. D. Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[19] Y. Jiang, Y. Xiang, X. Pan, K. Li, Q. Lv, R. P. Dick, L. Shang, and M. Hannigan, "Hallway based automatic indoor floorplan construction using room fingerprints," in *ACM UbiComp*, 2013.

[20] M. Mathie, "Monitoring and interpreting human movement patterns using a triaxial accelerometer," Ph.D. dissertation, The University of New South Wales, 2003.

[21] M. Susi, V. Renaudin, and G. Lachapelle, "Motion mode recognition and step detection algorithms for mobile phone users," *Sensors*, vol. 13, no. 2, pp. 1539–1562, 2013.

[22] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots*, vol. 15, no. 2, pp. 111–127, 2003.