Navigating in Signal Space: A Crowd-sourced Sensing Map Construction for Navigation

Jindan Zhu,[1] Souvik Sen,[2] Prasant Mohapatra,[1] and Kyu-Han Kim[2]

[1]Department of Computer Science,
University of California at Davis, Davis, CA 95616
{jdzhu, pmohapatra}@ucdavis.edu

[2]Hewlett-Packard Laboratories,
Palo Alto, CA 94304
{souvik.sen, kyu-han.kim}@hp.com

***Abstract*—Indoor navigation is typically achieved by an operational localization system among a range of location-based services it provides. However, the construction of a localization map, which is prerequisite for binding sensed observation in sensing space to individual locations in geographic space, remains a challenging task to date. In this work, we propose an indoor navigation system that alleviates the need for constructing a localization map and instead provides navigation in signal space. The main idea behind our approach is to construct a sensing map consisting of signal observations (WiFi Clusters) and connecting dead-reckoning segments obtained through mobile sensing capabilities (traces of accelerometer and digital compass reading). To this end, we design a prototype to demonstrate effective construction of such sensing map with energy-efficient sensors and crowd-sourcing, and its ability to support accurate navigation.**

I. Introduction

Navigation is a task of controlling the movement of subject from one location to another. Unlike localization, the objective of which is to position the subject in the physical or geographic space, navigation concerns more about the movement than the representation of end locations. Although pedestrian navigation is an indoor application that holds equal importance as localization, current navigation system is typically built upon an operational localization system. The source and destination locations are first localized in physical space, then the movement in between is inferred from the corresponding geographic layout such as a floor map. Such an approach limits the development of navigation as it heavily depends on the map of physical space, the *physical map*. The drawback is two-fold. First of all a physical map itself may not be readily available in practice, for administrative or security reasons. Furthermore, localizing with physical map requires building a *localization map* that projects observations in sensing space to their corresponding geographic locations. Producing such a binding has become one of biggest challenges for localization since it demands substantial amount of knowledge about both sensing and physical spaces. This process usually involves laborious and time consuming survey as adopted by various Wi-Fi fingerprinting and Simultaneous Localization and Mapping (SLAM) [1], [2] schemes. Recent work [3–6] managed to alleviate problems in map construction by exploiting unique features seen in both spaces. Despite the effort errors still build up when performing localization because the volatile nature of sensing space makes it too difficult to build stable association with the physical space without heavy investment. In the absence of a physical map and an accurate localization system, current navigation approach does not work.

Alternatively, if we refocus the navigation task to its essence about movement, we may find that the physical localization step is unnecessary. Navigation can be performed by following previous movement trails connecting two endpoints. A collection of these trails can create a general topological map in sensing space, the *sensing map*. In the sensing map a node represents a unique signal observation obtained at certain location, and the edge specifies moving displacement and direction between nodes. To navigate, the system first attaches signal observations at endpoints to the closest nodes on sensing map respectively, then works out the trail that consists of a sequence of intermediate nodes and connecting edges. By following the trail the user is navigated from one node to another according to the displacement and direction information, until the destination. Sensing map assumes no knowledge about the geographic space as locations are represented by signal observations and geographic layout is only inferred from sensed movement. Without the need to convert all elements back in geographic scope, a sensing map can avoid the dilemma of localizing and extending a localization map based on an inaccurate one, and its resulting accumulative error. Navigation task can therefore be completed independently and free from issues in an underlying localization system.

There are two intuitions supporting the construction of sensing map: 1) Geographic layout crucial for navigation can be largely preserved by movement information encoded as displacement vector between signal observations, and translated into plain instructions such as “walk forward for 10 meters to next intersection then turn right”; 2) Signal observations are statistically correlated with geographic locations. Therefore any location can be represented by one or more signal observations sharing certain level of similarity, which is the basis of many indoor localization systems. Practically the movement information is embodied as inertia sensor trace, and signal observation is collected through sensing ambient infrastructure such as Wi-Fi fingerprinting. Collection task of both can be crowd-sourced to users as they walk around their usual habitat. However putting crowd-sourced sensing information in use for the map construction still poses challenges:

1) Numerous studies have concluded that both of the sensed information are noisy and unstable, making it difficult to obtain unique signal observation for any location as well as accurate displacement vector between arbitrary nodes. In summary, Wi-Fi signal fluctuation resulted by indoor multi-path effect contributes a large part of localization error for the use of Wi-Fi observation, which is aggravated by diverse quality from crowd-sourcing due to device diversity and placement. Pedestrian dead-reckoning (PDR) technique has errors in estimating walking distance and angle from inertia sensor measurements. Such error can accumulate fast, creating a tendency of drifting away from actual trajectory as it prolongs.
2) Taking a crowd-sourcing approach suggests a problem of maintaining trade-off between collection efficiency, energy consumption, and sensor capability. Frequent and continuous use of energy-intensive sensors such as Wi-Fi radio should be limited, while in favour of low-power, ubiquitous sensors such as accelerometer. This requirement forces us to consider alternative to solutions such as detecting unique Wi-Fi signal landmarks through heavy Wi-Fi scanning.

Many solutions have been proposed to tackle these issues individually, and some of the techniques can also be incorporated into our scheme. However they are designed for localization systems thus inefficient for sensing map construction and navigation. After considering the characteristic of navigation task, we are particularly interested in one type of location with special geographic feature: the turns. A turn usually suggests a change in mobility whereas explicit for navigation task. It dissects complex mobility path into multiple segments within each the mobility is relatively uniform, thus can work as pivot points in a path. During navigation, giving instructions according to turns also provides user a useful visual clue for compensating small error in sensing map. Performing a turn is common and relatively easier to detect with energy-efficient inertia sensors, and provides a useful hint to collect more signal observations at turn locations for improving the robustness of fingerprint. Therefore constructing a skeleton map consists of accurate signal observations at turns and the displacement vectors connecting them is crucial for charting a full-fledged sensing map. In this process we battle the first challenge by clustering Wi-Fi observations and merging PDR segments from crowd-sourced data to boost stability. A modified density-based clustering algorithm is employed to group observations sharing signal similarity together as a cluster to represent a location. At turn locations where user path intersect frequently, dense observations can often be found thus result in better clustering accuracy and efficiency. Multiple PDR segments are merged if they are connecting same pair of turn clusters, or intermediate nodes in between. Measurement errors are averaged out, while drifting is minimized considering the relative displacement between clusters is short and simple. To achieve more energy efficiency, we also develop a scheme that can detect turns by looking at acceleration pattern, thus reduce the use of gyroscope sensor. At detected turn locations, the system collects Wi-Fi observations and magnetometer samples aggressively. Otherwise it switches to an opportunistic schedule with large interval for power saving.

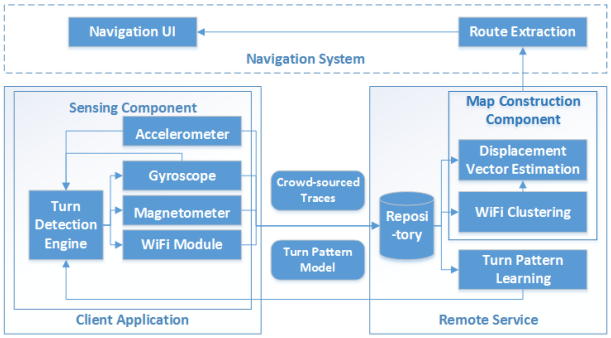

Fig. 1. System Architecture

In this work, we apply and test the principle of navigation with sensing map by developing a prototype that constructs a skeleton sensing map through autonomous crowd-sourcing. The main contribution of this paper is summarized as follows:

1) We propose separating the navigation task from localization task through the construction of a dedicated sensing map.
2) We design a crowd-sourcing friendly turn detection mechanism using discovered turn patterns from only accelerometer trace.
3) We adapt an iterative density based clustering algorithm to extract representative Wi-Fi observations.
4) We implement the system design on Android platform and evaluate its performance with real-life crowd-sourcing data.

The rest of this paper is organized as follows. In section II we give an overview of our proposed navigation system, and section III presents design and implementation details about turn-aware crowd-sourced trace collection, WiFi observation clustering and PDR displacement estimation. Section IV shows evaluation results on our proposed scheme. Section V provides background and state-of-art of WiFi localization and navigation, and finally we conclude this paper in Section VI.

II. System Overview

As shown in Figure 1, our current system consists of 1) a client-side application on smartphone responsible for harvesting and submitting crowd-sourced user trajectories and Wi-Fi observations, and 2) a remote service that learns turning pattern from training accelerometer trace, and processes user-contributed data into topological sensing map.

Most of the time the client runs in background sensing the user's mobility and environment. Its modules stay idle if user is static. When user starts walking normally, the client tracks his acceleration with accelerometer at 50Hz rate. Meanwhile it periodically triggers magnetometer and Wi-Fi scans to sense the walking direction and signal space at a lower 1/5 Hz rate. A turn detection engine is implemented to identify potential turns within accelerometer trace in real time, based on learnt turn pattern model. At the beginning of deployment the turn detection engine undergoes a training phase, in which accelerometer samples are annotated

when actual turn happens. The actual turn is obtained by gyroscope if available, or annotated by user manually. Training traces are sent to server where machine learning algorithm is applied to find the turn pattern. Whenever the turn detection engine identifies a turn, client immediately activates Wi-Fi module and magnetometer to collect Wi-Fi observations and change of direction at the turn location, and annotates them. While for the rest of the time they revert to the low sampling rate for power saving purpose.

Data collected by all participating users will be sent in a piggybacking manner to a central repository on which the remote service runs. The remote service carries out crucial processes of learning and map constructing. For each user, the turn pattern learning module builds a turn pattern classification model from annotated accelerometer trace for each user. Learnt model is returned to corresponding user client, or a generic model from multiple users is returned if no training trace is provided. The Wi-Fi clustering component performs density based clustering on entire Wi-Fi observation dataset to extract similar Wi-Fi observations encountered by different users during their walks, preferably at the same turn they passed through. When new cluster is found, the displacement vector estimation component calculates a path to other existing clusters by fusing dead-reckoning traces that connect them. As a result, a topological sensing map made of Wi-Fi clusters and displacement vectors is constructed incrementally. On top of this system, a navigation module can be later built and plan route based on the constructed sensing map.

III. System Design

In this section we describe in detail the design of major components in our system, motivation to each of these components and their validations.

A. Turn Detection

Turn location plays a significant role in two regards: navigation and sensing. Most indoor environments maintain a layout consisting of open spaces and structured pathways. Typical navigation task that entails finding a pathway connecting two spaces can be conveniently restated as finding a series of path segments pivoting at turns. Users usually cope well with the by-turn instructions given that they can perceive clues from surrounding layout. On the other hand, movements within each path segment are largely constrained by its monotonous structure, simplifying their accurate extractions from the noisy and under-sampled inertia sensor trace. Only at turns the user mobility is forced to change and thus exhibits a change of pattern in acceleration, which is relatively frequent and easy to detect compared to other landmarks. Based on these considerations, we propose constructing our sensing map out of a skeleton network of turns, in accordance with the nature of navigation. Compared to ordinary locations, Wi-Fi observations are collected aggressively at turn locations in order to increase the effectiveness of finding corresponding Wi-Fi clusters and the amount of connecting PDR segments.

The design hinges on detecting turns in an energy-efficient and accurate way. Cumulative integration on gyroscope samples is known to be simple and effective in detecting turn. However working on the gyroscope requires continuously monitoring additional sensor, costing more power (6mA on Samsung i9500 smartphone, compared to 0.25mA of the accelerometer). Furthermore gyroscope may not be available on majority of inexpensive devices, as currently out of more than 1500 listed devices only 304 are equipped with a gyroscope [7]. To complement this situation we also develop a solution to detect turns based on accelerometer samples only. Accelerometer is essentially ubiquitous on all handheld devices, and it is always-on anyway for supporting PDR and other applications. By studying subjects walking at average speed and making turns, we observe that a typical turn is most likely to be finished within three steps, and takes less than 1.5 seconds. The movement can be dissected into a sequence of consecutive linear accelerations along mediolateral axis biased towards the direction of the turn. In addition as body trying to stabilize during the turn, magnitude of movement on both superior-inferior and anteroposterior axis reduces compared to walking straight. If 3-axis accelerometer can capture these subtle changes in movement, it is possible to detect turns from accelerometer trace.

This component consists of two parts: the sensing component collects mobility related sensor traces and detects turns based on learnt model, and the learning component mines the patterns from training traces. We apply machine learning technique to extract the potential turn pattern from accelerometer trace. We use gyroscope detected turns as ground truth to annotate the accelerometer trace for training. A supervised classifier is built from the training trace.

1) Feature Extraction: We adopt a half-sliding-window on all three axes for extracting features of the window. Since the sampling rate is 50Hz we choose a window size of 64 so that a window will contain 1.28 seconds of samples, roughly corresponding to two to three steps during a walk and is enough time to complete a turn. The window moves for every 32 samples to limit the detection delay. For the windowed samples, a total of 18 features are calculated and summarized in Table I. In the table $N = 64$ is the window size. $k, l \in x, y, z$ denotes the axis of trace. a_n^k denotes the n^{th} sample in windowed trace of axis k. In the energy calculation, the samples are first processed by a low-pass filter with cut-off frequency empirically set at 6Hz, which no walk related movement is likely to exceed. The FFT window size is the same as the sliding window. All feature calculations can be implemented efficiently on smartphone and performed in real-time without incurring too much computational overhead.

2) Axis-Movement Association: Before we start training and using learnt model to detect turns in real-time, we need to make sure the acceleration axes of training and testing traces follow the same coordination system. Due to the ambiguity of phone orientation, we need to determine which axis of acceleration corresponds to what direction of movement. Solution to this problem has been studied [8]. However since exact orientation is not required but a rough association of axes in our system, a simple scheme is employed to identify the axis-movement association, and it performs well throughout the turning experiments. We

TABLE I
LIST OF EXTRACTED FEATURES

Name	Description
Arithmetic Mean μ^k	$\mu^k = \frac{1}{N}\sum_{i=n}^{N} a_n^k$
Standard Deviation σ^k	$\sigma^k = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(a_n^k - \mu^k)^2}$
Range R^k	$R^k = (Max(a_n^k) - Min(a_n^k))\|_{n=1\ to\ N}$
Inter-axis Correlation $\rho^{k,l}$	$\rho^{k,l} = \frac{\sum_{n=1}^{N}(a_n^k-\mu^k)(a_n^l-\mu^l)}{\sigma^k\sigma^l}$
Inter-window Correlation $corr^k(t-2,t)$	the correlation between two consecutive non-overlapping window.
Spectrum Energy S^k	$S^k = \left\|\sum_{n=1}^{f_{cutoff}} x_n e^{-2\pi i f n}\right\|^2$,

examine the frequency components of windowed samples from three axes using FFT. The axis produces largest DC component is associated with vertical movement as it is influenced most by gravity. Then the scheme searches through the spectrum of identified superior-inferior axis, and finds the frequency at which the maximum energy is observed. This frequency is denoted as *steps frequency*, for it reflects the repetitive pattern of walking. Note that the anteroposterior movement coincides with the same repetitive pattern therefore normally has its maximum energy component observed at the same frequency. And we choose the axis that has lowest energy at this frequency among three as lateral axis. By doing so, an association between phone's current orientation and the one used to describe walking direction is created, and later used to match features in training trace to that of testing trace. This association is checked on a per-window basis. If association changes are reported three consecutive times the system believes a change of association occurs and waits for required amount of samples to update to new association. Any turn detected during this period are discarded until new association stabilized.

3) Turn Pattern Learning: After collection of training traces and extraction of their features, we normalize the resulting traces of features and input them to a neural network for a training of supervised classification. For an instance, each feature of the windowed samples is treated as one attribute to the classifier. The binary class attribute of ground truth is obtained by integrating gyroscope samples in the same window duration. Output of the training is the turn pattern model we can later use to detect the turns on client side. The learning component is implemented using Java Machine Learning Library (Java-ML) [9] and its Weka [10] interface.

To validate our intuition that turn patterns do exist in accelerometer trace, we perform several experiments to assess the effectiveness of the learning. We use a 30×25 square shaped corridor for the walking/turning experiment. In each trail we ask the participant to walk through the corridor for 3 rounds, thus making 11 turns. The phone is held in three typical positions: reading in *hand*, *belt* holster, and pants front *pocket*. For each position a left turn trail and a right turn trail are made. Three participants conducted the experiment in different days, and we collected 18 trails and 198 turns in total. We generate the feature traces from collected accelerometer traces as described in previous sections, and feed them to the learning algorithm. Traces are randomly subsampled with a SpreadSubsample filter to maintain a 1/3 proportion of turn/non-turn to mitigate imbalance sampling problem. Each trace is re-subsampled 10 times and re-learnt and their average result is reported. For learning we use a two layer MultilayerPerceptron classifier with default parameters and test its performance with 10-fold cross-validation. The effectiveness of learnt model can be measured by precision and recall metrics for each class.

Among them we are particularly interested in the recall for turn class and the precision for non-turn class, since we have a goal to minimize missed detection of actual turn while a few false positives in turn class are tolerable.

First we examine the overall performance regardless position. For each participant all traces are used for the learning. Figure 2 shows the result. We can see that there is indeed a pattern of turn within user's acceleration. With a small period of training, we can discover the pattern with a reasonable accuracy.

We also look at the effect of different holding positions on the pattern discovery. As illustrated in Figure 3, both hand and belt position give relatively good result while in pocket position the pattern is obscured. After closer analysis of the traces we believe the reason is that when the phone is in pocket, the rotational movement of the legs during regular walking generates substantial amount of noisy acceleration which makes it more difficult to be distinguished from those of a turn. This situation is less severe when the phone is attached to the body trunk that is more stable.

When we look into the individual feature trace we also find some interesting results. Some features show more relevance than others. Specifically coinciding with the time when turn happens, we observe dips in the energy on vertical axis, and the inter-window correlation on both vertical and anteroposterior axis. Figure 4 shows a typical example of these three features with respect to turns. In the example Y-axis is the vertical axis and Z-axis is the anteroposterior axis. Traces are manually layered to increase visibility. To our surprise, features on the lateral axis does not consistently exhibit changes as visible. We believe there are two reasons: 1) sometimes participants prefer a pattern that breaks down a turn into multiple step so that the lateral acceleration becomes too small to notice. 2) the noise from leg movement when phone is placed in pocket.

B. Iterative Density-based Wi-Fi Clustering

After gathering the crowd-sourced Wi-Fi observations and PDR traces from user, the system needs to determine representative observations for waypoints along a trace. Since indoor Wi-Fi signal fluctuates from time to time, observation at the same location may vary. Despite this instability, Wi-Fi observations collected at nearby location still show a distinguishable level of similarity suggesting correlation between signal similarity and physical distance, as reported in various Wi-Fi fingerprinting studies. By controlling the signal similarity level, we can find from crowd-sourced data the group of Wi-Fi observations representing a finely confined area, or simply the same location. Although

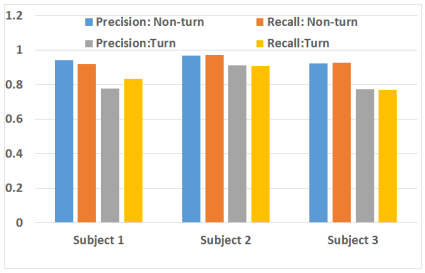

Fig. 2. Turn Classification Accuracy: Subject

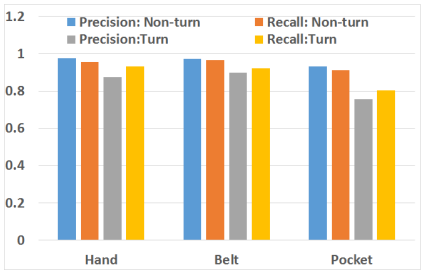

Fig. 3. Turn Classification Accuracy: Position

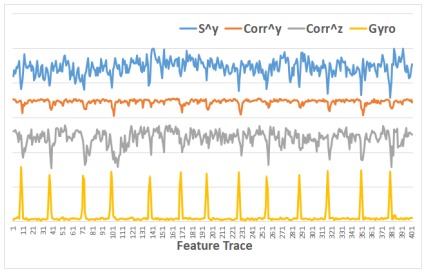

Fig. 4. Turn Classification Accuracy: Feature

conversely, a same location is allowed to have multiple groups to represent it, due to the inevitable signal variation. We use a clustering algorithm to find these groups. Number and quality of found Wi-Fi clusters are crucial for the accuracy of merging PDR segments connecting them, and ultimately the construction of sensing map.

Turns along a trail serve as better waypoints than regular locations. Since we collect Wi-Fi observations more frequently at turn locations, the sample distribution will be denser near turn locations over other places. This naturally leads us to select a density based clustering algorithm for the clustering task. Besides its property of handling noise/outliers well, density based clustering has the ability of finding arbitrary shape clusters which also makes it more desirable over other techniques, as our Wi-Fi observations are collected along a trace while users are mobile and uncoordinated. Our implementation is based on DBSCAN [11] clustering algorithm with several adaptations. Following sections describe the procedure in detail.

1) Wi-Fi Observation and Signal Distance: First we make a brief note about a few technical problems in our Wi-Fi observation collection, and define the distance metric for clustering. Channel diversity enjoyed by nowadays Wi-Fi chipsets adversely poses difficulty to our data collection. For instance for a device supporting both 2.4GHz and 5GHz bands the full scanning process can take up to 5 seconds to complete, which is unacceptable when user is mobile. To address this issue we made a trade-off by limiting the scans to ten channels in two bands which AP most commonly operates on. The time to finish one scan is reduced to about 1 second, and all scans will have uniform set of channels. During our data collection we find numbers of virtual APs are set up to broadcast multiple BSSIDs. These virtual APs are in fact co-located and operate on same channel thus have the same signal strength. Treating them as unique APs when calculating the signal distance will bias the result. Therefore we group these virtual APs into single AP during the calculation.

We use Tanimoto coefficient as the similarity metric to measure signal distance between two Wi-Fi observations. Let obs_i and obs_j be two observation vectors of Wi-Fi AP-RSSI pairs, the signal distance between them $SDist(i,j)$ is defined as:

$$SDist(i,j) = 1 - \frac{obs_i \cdot obs_j}{|obs_i|^2 + |obs_j|^2 - obs_i \cdot obs_j}$$

If an AP only appears in one of the observations, the absent signal is set to value zero for calculation. A constant of -95dBm is subtracted from all RSSI absolute values to reflect the correct effect of signal strength in the presence of absent APs.

2) Iterative Clustering and Pruning: The generic DBSCAN algorithm controls the clustering process through two parameters, a distance threshold $\varepsilon - distance$ and a density threshold *minPts*. If point p has a distance less than ε to point q, p is in the $\varepsilon - neighborhood$ of q and vise versa. If p has a dense $\varepsilon - neighborhood$ with members more than *minPts*, it becomes a *core*. DBSCAN defines a model of density reachability stating that q is *density-reachable* from p if there is a chain of points $p_1, ..., p_n$ where $p = p_1$, $q = p_n$, and p_{i+1} is in the $\varepsilon - neighborhood$ of core p_i $(1 \leq i \leq n-1)$. p_1 and p_2 are defined *density-connected* if there exist a point q so that p_1 and p_2 are density-reachable from q. The clustering is a process of merging small dense areas centered at core into a larger area, by grouping all density-connected points into the same cluster. The algorithm starts with a randomly chosen point and repeatedly looking for cores and their density-connected points to form clusters.

In our scheme we use $\varepsilon - distance$ threshold for controlling the compactness of a cluster, so that the cluster can be representative for a small area. A minimal membership of cluster is also required so that the property of cluster is stable and not easily swayed by wrongly clustered outliers. This is controlled by *minPts* parameter. Density reachability allows DBSCAN to find clusters of arbitrary shape, however also tends to cause the clusters to over-grow. Therefore we make following adjustments to prune the over-grown clusters. We define another compactness threshold δ. If a cluster has compactness exceeds δ, we mark it as over-grown and is to be split. At each stage of splitting, all clusters marked over-grown together with all the noise points will form a new data set. A DBSCAN clustering operates on the new data set with increased *minPts* by one per stage, to focus on the denser portion of over-grown clusters while exclude the sparse part. This process is repeated iteratively for at least five times and stops until there is no new cluster can be formed in two consecutive iterations.

When a new observation is added incrementally, the scheme will calculate its $\varepsilon - neighborhood$ and mark affected existing clusters to be updated if they have members in the new observation's $\varepsilon - neighborhood$. New round of clustering is performed on affected clusters and noise. If no new cluster is formed then existing clusters are kept and the new observation is marked noise.

3) Clustering Parameters: The value of $\varepsilon - distance$ and threshold δ are the most important parameters concerning the clustering, since they determine the compactness of resulting clusters. Choice of the values is based on the signal-

spatial correlation that smaller signal distance indicates nearby locations. To verify this signal spatial correlation, we first examine the signal characteristics. We use the same experiment setting as in turn pattern learning. Wi-Fi observations are collected by participant walking along the trails, in which coordinates of turn location observations are manually marked afterwards and other observations' coordinates are interpolated. We calculate the pairwise signal distance as well as physical distance for all samples, to find out how do they correlate.

First we examine the extent of signal variation at different locations. Signal distance samples for observation pairs collected at same location are isolated and the CDF of which is plotted in Figure 5. From the result we can see signal distance varies a lot even when collected at the same location. However over 90% are under 0.2, which provides us a guideline for choosing $\varepsilon - distance$ and δ within this range.

Next we look into the signal-spatial correlation. We calculate physical distance from all observation pairs that with signal distance less than 0.2. We modify the signal distance threshold by a 0.02 step, and see the distribution of physical distance among all qualifying samples within the cutoff signal distance. We plot their mean, standard deviation, and size in Figure 6. Smaller the physical distance means better compactness of samples within the cutoff signal distance. The result shows a correlation between physical and signal distance, suggesting it is plausible to use signal distance parameters to control the physical compactness of clusters. However the available samples also drop fast as signal distance declines. To make a trade-off, we choose 0.12 as $\varepsilon - distance$ to form nucleus of clusters, and set δ to 0.16 to regulate the final compactness of formed cluster.

Parameter *minPts* can affect the effectiveness of finding clusters given the dataset. The optimal value of *minPts* may change as the size of total dataset increases. From Figure 6 we find about 20% of samples are under $\varepsilon - distance$ of 0.1 and there are at least four turn locations. Therefore we choose empirically a relaxed *minPts* around 3% of the dataset size. We will evaluate the effect of different parameter choices in next section.

C. Displacement Vector Estimation

Once clusters representing waypoints along the trace are found, we can determine the displacement vector between them through a PDR fusion process. Each Wi-Fi observation is bind with a sample in the corresponding PDR trace by matching the timestamps. Given two clusters, the displacement estimation component examines their members by pairs. If a pair of members have associated PDR samples belong to the same PDR trace, the segment in between is extracted and checked whether it is a straight segment. If any sample in the segment is detected as a turn, this segment is discarded. After all pairs of members are accounted for, the displacement estimation component fuses all straight segments into a single displacement vector by averaging the distances and angles.

Extracting displacement from a PDR trace has been heavily studied in the past[4], [12], [13]. Although sophisticated solutions certainly can achieve better performance, a basis approach involving step counting and frequency-based stride length estimation is incorporated. Same technique described in [14] is used for counting steps. Magnitude of acceleration trace is passed into a low-pass filter to remove high frequency noise that cannot be associated with steps. Derivative of the resulting output is calculated and the crossing point when its value reverses from positive to negative suggests a local maximum, thus a step. Frequency-based stride length estimation [15] model the linear relation between stride length and step frequency as $stride = a \times freq_{step} + b$. In evaluation we assume the parameters a and b are trained beforehand.

Although our scheme relies more on turns than the actual turning angle, angle reference is still a useful clue especially when a navigation starts. In our data collection we observed that the values of both absolute direction and relative turning angle are highly unreliable and misleading. However readings collected within a straight segment are fairly consistent throughout different walks, which makes it possible to hint the direction of a segment. There are several techniques[8], [16] for estimating initial orientation offset of device, however in this work we do not emphasize this and assume it is known and will not drastically change during a walk. If any sample in a PDR segment possesses a compass reading, an angle reference is added to this segment. If large conflicting compass reading appears in a segment, it may suggest an undetected turn or simply magnetic turbulence. This segment is temporarily removed until more data received can confirm if there is a missed turn. When multiple segments are fused into one displacement vector connecting two clusters, all available angle references are also averaged to represent the direction of the displacement vector.

IV. Evaluation

Based on realistic traces collected through experiments, we evaluate the feasibility and effectiveness of our proposed system.

A. Experiment Setup

We collect trace data inside a campus building where offices, computer labs, and classrooms locate and with moderate populace and traffic. Users walk around the main corridors carrying smartphones equipped with user client. For each walk user carries the phone in one of the three positions described previously, and with no significant change in phone orientation. All inertia sensors record at a rate of 50Hz, and the Wi-Fi scan is performed at 1/5 Hz for non-turn locations. We use gyroscope samples to mark the ground truth of turns, so as to limit the interference of user intervention. Although in reality users can start and stop a route at any point and in any direction, to simplify the coordinates assignment of ground truth locations we use one designated route. The route is shown in Figure 8 where red arrow indicate the path and direction, and red start and octagon mark the start and end respectively. The route is connected by 8 turns. Each turn is assigned a coordinates manually during analysis, and for samples collected between turns their coordinates are interpolated according to time intervals. At normal walking speed it

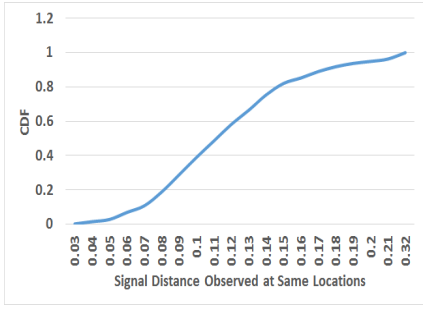

Fig. 5. Signal Variation at Same Locations

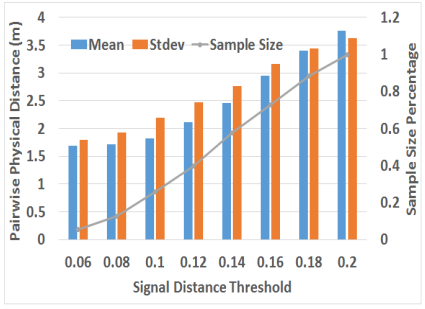

Fig. 6. Signal-Spatial Correlation

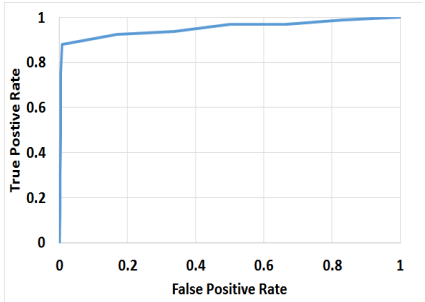

Fig. 7. ROC Curve of Turn Detection

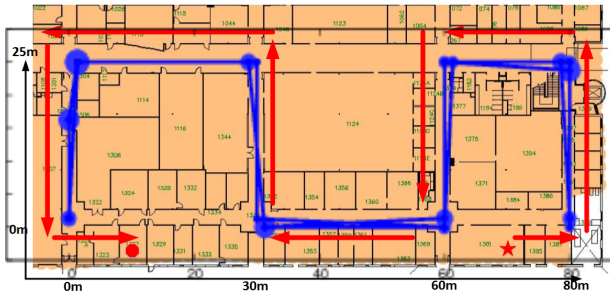

Fig. 8. A Sensing Map

takes about three minutes to walk through the route. We collect a total of 30 traces for the route over a seven-day period.

B. Evaluation Metrics

Using this dataset we evaluate both the turn detection accuracy, as well as the clustering performance and sensing map accuracy.

When measuring the performance of turn detection we are interested in its *precision* and *recall*.

$$Precision = \frac{TP}{TP+FP},\ Recall = \frac{TP}{TP+FN}.$$

To evaluate the general performance of the clustering algorithm, we adopt the standard *BCubed precision* and *BCubed recall* metrics with respect to the ground truth:

$$Precision\ BCubed = \frac{\sum_{i=1}^{n} \frac{\sum_{o_j: i\neq j, C(i)=C(o_j)} Correctness(o_i,o_j)}{|o_j|i\neq j, C(i)=C(o_j)|}}{n}$$

$$Recall\ BCubed = \frac{\sum_{i=1}^{n} \frac{\sum_{o_j: i\neq j, L(i)=L(o_j)} Correctness(o_i,o_j)}{|o_j|i\neq j, L(i)=L(o_j)|}}{n}$$

, where n is the size of dataset. $L(o_i)$ denotes the ground truth category of object o_i, $C(o_i)$ is the cluster object o_i is classified to. Correctness is defined as:

$$Correctness(o_i,o_j) = \begin{cases} 1 \text{ if } L(o_i)=L(o_j) \Leftrightarrow C(o_i)=C(o_j) \\ 0 \text{ otherwise} \end{cases}$$

Accuracy of sensing map can be measured by the quality of found clusters, and the error in estimated PDR segments. For each individual cluster we are most interested in *physical compactness*, defined by averaging the pairwise physical distance $a(o)$ for all o in the cluster C_i: $a(o) = \frac{\sum_{o' \in C_i, o\neq o'} dist_{phy}(o,o')}{|C_i|-1}$. Cluster *centroid error* is another important metric, defined as the Euclidean distance between coordinates of signal centroid and actual physical centroid. PDR distance error is calculated with estimated inter-cluster distance and the ground truth distance between physical centroids of corresponding clusters.

C. Evaluation Results

1) Turn Detection Accuracy: We use turn pattern model trained from experiment in the previous section to predict turns in this dataset. The training and testing set are collected under different circumstances, and only part of the turn locations appears in both dataset. When we looked into the raw result reported by the classifier we found degraded performance as both metrics dropped to about 70%, which after analysis is caused by the slightly out of sync between the windowed features of prediction and ground truth. Instead of examine precision and recall at per window level, we manually inspect the result at turn level by defining that a turn is correctly detect if it is within 2 windows from the ground truth. The time difference is less than 1.28 second which we consider acceptable delay without causing large error in Wi-Fi collection. We reach a turn-level detection precision of 0.812, and recall of 0.895. The ROC curve of the detection is shown in Figure 7, the large area under the curve indicates a high accuracy.

2) Accuracy and Coverage: We use $\{\varepsilon - distance = 0.1, \delta = 0.16, minPt = 11\}$ on whole dataset and evaluate the overall performance. Figure 9 shows the CDF of centroid error in found clusters. We observe an average error of 0.72m with 90% of errors less than 1.5m. It indicates strong signal-spatial correlation at locations represented by clusters. Physical compactness result in Figure 10 shows 90% of errors less than 3.5m and the average error is 2.3m, suggesting compact clusters formed are able to uniquely represent locations.

Figure 11 illustrates the error in inter-cluster distance estimated by fusing PDR segments. The average error is 2.23m, and a 90 percentile at about 4.5m. The error is small enough for human to correct automatically with a few environmental cue. Also the estimation can be further improved with more sophisticated PDR algorithms. Although our system replies primarily on turns for determining relative direction in navigation instead of absolution orientation, the orientation information obtained from magnetometer can still provide a coarse clue in helping user adjust the initial heading and determining if it is on correct path. Individual angle collected during the walk exhibits large error thus lack of practical use. However surprisingly after fusing PDR segments the orientation reading shows certain level of consistency along same direction. The result is summarized in Table II.

We examine the effect of amount of data trace as they incrementally added to the system. Portion of the data set is

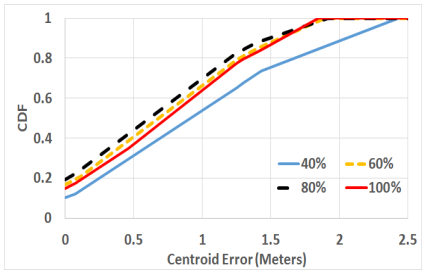

Fig. 9. Error in Cluster Centroid

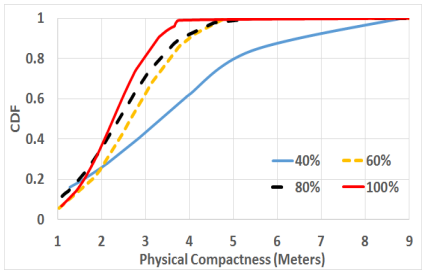

Fig. 10. Cluster Compactness

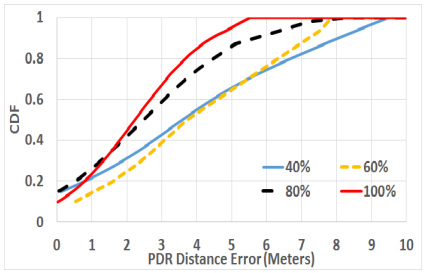

Fig. 11. Error in PDR Distance Estimation

TABLE II
FUSED PDR DIRECTIONS

Source	Destination	Direction Mean	Direction Deviation
(0,25)	(0,0)	159.2	18.07
(30,25)	(0,25)	246.6	11.39
(30,0)	(30,25)	300.4	27.18
(60,0)	(30,0)	241.5	8.95
(60,25)	(60,0)	164.2	10.27
(80,25)	(60,25)	257.7	11.03
(80,0)	(80,25)	298.8	29.78

randomly sample from 40%. As expected accepting more data will gradually improve accuracy in clustering and PDR estimation, as shown in Figure 9 to Figure 11. Coverage related statistics is plotted in Figure 12. As size of data set increases, more clusters are formed. We found for most cases all turn locations are covered with at least one cluster, except for one missing in 40% data. This result proves the effectiveness of turn detection scheme. Average member size of cluster increases as more data coming in, resulting a higher number of PDR segment fused. Especially with 80% of the data set, all the seven non-overlapping PDR segments between turn locations are covered and a connected sensing map is formed.

3) Effect of Clustering Parameters: First we examine the effect of varying $\varepsilon - distance$. We fix *minPt* value at 11 and set δ to a large value 0.26, then modify $\varepsilon - distance$ from 0.06 to 0.2 at a step of 0.02. Figure 13 shows the change in performance of clustering in terms of BCubed precision and recall, and the number of clusters found. When $\varepsilon - distance$ is small the high precision suggest formed clusters are pure, in contrast to the low recall because fewer clusters is found to accommodate many the samples collected at the same location. As the $\varepsilon - distance$ increases, cluster becomes more and more heterogeneous, resulting in lower precision but high recall. The number of cluster increases up to a point then reverse due to previously compact clusters are merged into a few large clusters.

Next we fix $\varepsilon - distance$ to 0.1 and δ to 0.16, and change the *minPt* value from 8 to 20, in a range of about 1% to 3% of the data set size. Figure 14 shows the change in BCubed precision and recall, and the number of clusters found. *minPt* has a large impact on the number of clusters found, but not on the BCubed metrics. It only influences the members of noise cluster, causing slight fluctuation in the BCubed metrics.

D. Discussion

The evaluation shows promising accuracy for the turn detection engine, nonetheless we believe there is still room for improvement. The difference in classification accuracy for three positions suggests that phone placement will affect the turn pattern in various ways. By setting up dedicated pattern model profile for each phone placement in prediction, it is possible to further enhance the detection robustness. We plan to study in future the run-time phone placement identification for more commonly observed placements, and apply corresponding turn model for identified placement.

Energy awareness is one key incentive for successful crowd-sourcing systems. Through reducing the activation of power hungry radios while relying on low energy inertial sensors, the construction of sensing map based on turn locations demonstrates the feasibility of enabling navigation without incurring high energy consumption to participating devices. Although our clustering scheme is based on Wi-Fi fingerprinting, it is straightforward to adopt other types of signal observation for representing nodes in sensing space. For Wi-Fi fingerprinting there exists many solutions in order to improve the crowd-sourced collection by dealing with common issues such as device diversity or temporal variance. In our future work, we are also interested in integrating these schemes into our system, and evaluating a full-fledged crowd-sourcing solution in a large scale setting with more participants, devices diversity, and environment diversity.

V. RELATED WORK

WiFi fingerprinting based localization [17–19] recognizes the statistical correlation between WiFi RSSI signal vector and the location where it is observed. Such bindings can be collected and compiled into a localization map for interested space, usually through a dedicated or crowd-sourced survey. This map can later be used to localize user who observes a new WiFi signal vector by matching the vector to existing one on the map. This approach takes great advantage of ubiquitous WiFi infrastructure indoor and RSSIs easy-to-extract property from the multiplying WiFi enabled commercial mobile devices. Nonetheless such a localization map based approach requires surveying at known locations, which seriously undermines its practicality. Therefore we adopt its principle on WiFi fingerprint as a representation to location but seek other ways to remove the constraints in sensing map generation.

Techniques from simultaneous localization and mapping (SLAM) [20], [21] have been adapted into efforts to enable the automatic localization map generation. WiFi landmarks [1], [2], [22] are placed and can be perceived through sensing. In localization user localizes himself with respect to a landmark, and then as he moves and encounters previously learned landmarks their locations with respect

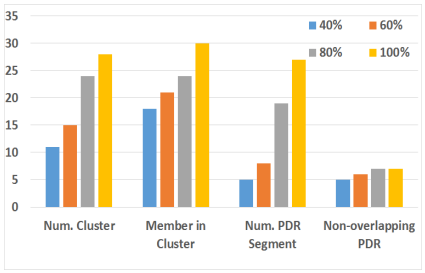

Fig. 12. Coverage Statistics

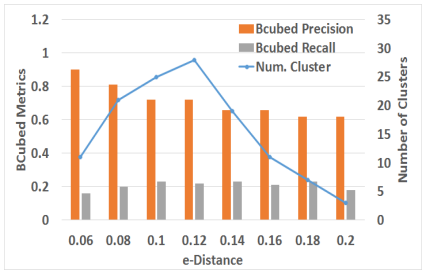

Fig. 13. Effect of e-Distance

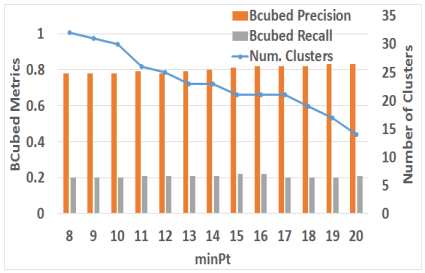

Fig. 14. Effect of minPt

to each other can be determined by the mapping. These schemes and their applications [23], [24] have demonstrated the feasibility of tracking an existing PDR trace. SLAM approaches treat localization and mapping as recursive, inter-dependent processes. Several stochastic process models have been created to handle the signal fluctuation and noisy dead-reckoning trace. Whereas in our approach we separate mapping from localization process. Landmarks are extracted in parallel from large crowd-sourced data by employing a clustering technique that exploits the signal similarity and stability at some of the locations, and only their locations relative to each other are of interest and preserved by fusing the PDR trace.

Several recent work also proposes ways that help better estimate landmark locations by identifying special landmarks that exhibit unique physical features or sensing features. In LiFS [5] and Zee [4] a floor plan is assumed to extract some physical features that can aid regulating the sensed landmarks. Special types of signal landmarks are discussed. In unLoc [3] WiFi observation with least similarity to nearby observations are used as a landmark. In Walkie-Markie [6] a WiFi-Mark is identified as a point where WiFi signal from an AP reverses its trend. In our approach landmarks are WiFi observations that can be clustered from crowd-sourced data. We pay special attention to landmarks collected at turns from the point of view of navigation task and energy efficiency in crowdsourcing. Different from aforementioned signal landmarks that usually require frequent WiFi scanning to detect, change of mobility pattern at turns can be detected through low-power accelerometer alone and works as trigger for proactive WiFi landmark collection at turns which improves both clustering and power efficiency.

VI. Conclusion

By recognizing the different nature of navigation from localization, we propose the construction of a sensing map to facilitate the deployment of an indoor navigation system. A sensing map, parallel to localization map, is more suitable to create by a crowdsourcing system in an automatic and energy-efficient manner. Based on the intuition, we design and implement the system generating the sensing map consists of WiFi clusters and connecting PDR displacement vectors. Preliminary results demonstrate the effectiveness and accuracy achieved by proposed system and resulted sensing map, and potential for the successful conducting of navigation tasks.

References

[1] B. Ferris, D. Fox, and N. Lawrence, "Wifi-slam using gaussian process latent variable models," ser. IJCAI'07, 2007, pp. 2480–2485.

[2] H. Shin, Y. Chon, and H. Cha, "Unsupervised construction of an indoor floor plan using a smartphone," vol. 42, no. 6, 2012, pp. 889–898.

[3] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: unsupervised indoor localization," ser. MobiSys '12, 2012, pp. 197–210.

[4] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," ser. Mobicom '12, 2012, pp. 293–304.

[5] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," ser. Mobicom '12, 2012, pp. 269–280.

[6] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-markie: indoor pathway mapping made easy," ser. nsdi'13, 2013, pp. 85–98.

[7] PhoneArena. (2014) Phone finder and compare. [Online]. Available: http://www.phonearena.com/phones/full#/phones/full/?f[325][]=1600

[8] D. Mizell, "Using gravity to estimate accelerometer orientation," in *Wearable Computers, 2003.*, 2003, pp. 252–253.

[9] JavaML. (2014) Java machine learning library. [Online]. Available: http://java-ml.sourceforge.net/

[10] Weka. (2014) Weka 3: Data mining software in java. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/

[11] M. Ester, H. peter Kriegel, J. S, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." AAAI Press, 1996, pp. 226–231.

[12] U. Steinhoff and B. Schiele, "Dead reckoning from the pocket - an experimental study," ser. PerCom '10, 2010, pp. 162–170.

[13] S. Sprager and D. Zazula, "Impact of different walking surfaces on gait identification based on higher-order statistics of accelerometer data," ser. ICSIPA '11, 2011, pp. 360–365.

[14] R. Libby. (2008) A simple method for reliable footstep detection in embedded sensor platforms. [Online]. Available: http://ubicomp.cs.washington.edu/uwar/libby_peak_detection.pdf

[15] D.-K. Cho, M. Mun, U. Lee, W. Kaiser, and M. Gerla, "Autogait: A mobile platform that accurately estimates the distance walked," ser. PerCom '10, 2010, pp. 116–124.

[16] K. Kunze, P. Lukowicz, K. Partridge, and B. Begole, "Which way am i facing: Inferring horizontal device orientation from an accelerometer signal," in *ISWC '09.*, 2009, pp. 149–150.

[17] P. Bahl and V. N. Padmanabhan, "Radar: an in-building rf-based user location and tracking system," 2000, pp. 775–784.

[18] M. Youssef and A. Agrawala, "The horus wlan location determination system," ser. MobiSys '05, 2005, pp. 205–218.

[19] J.-g. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie, "Growing an organic indoor location system," ser. MobiSys '10, 2010, pp. 271–284.

[20] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (slam): Part i the essential algorithms," *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, vol. 2, p. 2006, 2006.

[21] P. Robertson, M. Angermann, and B. Krach, "Simultaneous localization and mapping for pedestrians using only footmounted inertial sensors," in *In Proc. UbiComp 2009, ACM*, 2009, pp. 93–96.

[22] L. Bruno and P. Robertson, "Wislam: Improving footslam with wifi," ser. IPIN '11, 2011, pp. 1–10.

[23] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, "Did you see bob?: human localization using mobile phones," ser. MobiCom '10, 2010, pp. 149–160.

[24] H. Shin, Y. Chon, K. Park, and H. Cha, "Findingmimo: tracing a missing mobile phone using daily observations," ser. MobiSys '11, 2011, pp. 29–42.