A Learning-Based Approach towards Localization of Crowdsourced Motion-Data for Indoor Localization Applications

Farhang Vedadi
Electrical & Computer Engineering Department
University of Toronto
Toronto, Canada
farhang.vedadi@mail.utoronto.ca

Shahrokh Valaee
Electrical & Computer Engineering Department
University of Toronto
Toronto, Canada
valaee@ece.utoronto.ca

基于指纹的室内定位?

***Abstract*—Many popular fingerprinting-based indoor localization methods, such as WiFi-based localization systems, rely on a dataset of fingerprints labelled by known locations (fingerprint-location dataset) to be able to localize a user by comparing the queried fingerprint with the fingerprints in the dataset. Generating and updating such a dataset is a burden and requires significant amount of time, human effort and expertise. In this work, we propose a system to build such a dataset from scratch using crowdsourced data. Consequently we reduce the required time and effort, by leveraging the information shared by the crowd. The proposed system is based on localization of user motion, hence called LocaMotion. LocaMotion takes a supervised learning perspective towards localization of user-sent motion data. In other words, LocaMotion is formalised as a classifier that extracts and assigns features from a user motion data to a sequence of points in the building. Training and testing procedures for LocaMotion will be discussed in details followed by extensive experiments to demonstrate its validity and success to localize motion patterns and generate useful fingerprint-location datasets.**

***Index Terms*—Indoor Localization, Crowdsourcing, Supervised Learning, Motion Patterns.**

I. Introduction

Indoor location estimation, commonly referred to as indoor localization, has been an active research in recent years [1], [2]. In a WiFi fingerprinting-based system, the localization problem is cast as the problem of relating the WiFi received signal strength (WRSS), referred to as a fingerprint, and the location of the user. To estimate this model, an expert has to collect a sufficiently large number of WRSS samples in different locations of a building. Then, different learning algorithms are applied with the collected WRSS fingerprints as features, and their associated locations in the building as labels. The result is a model that can predict the location (label) of a future queried WRSS fingerprint (feature) [3]–[6].

The dataset of WRSS and location used for training plays an important role in the accuracy of the location estimations. However, building and maintaining such a dataset is very challenging. First, collecting a sufficiently large number of WRSS samples from visible WiFi access points at known locations in a building can be a time-consuming task and requires a considerable amount of time, human effort and expertise. Second, indoor environments are subject to changes that can make portions of the dataset obsolete. It is important to be able to update this dataset occasionally to correct for changes in the visible WRSS values. Considering the burden of frequently creating/updating the WRSS-location dataset by an expert, a new trend for designing systems that can use other sensors and/or crowd of users to build/update the WRSS-location dataset has been created [7]–[9]. These systems are referred to as *automatic labelling/fingerprinting* systems or *crowd-sourcing-based localization systems*. The key idea in a crowd-sourcing-based method is that users can potentially share useful sensory information such as acceleration and/or angular velocity measured by their smart devices. Then, this information can be used to infer the location of the user-sent WiFi-fingerprint and to create/update the dataset of WRSS-location. Some approaches have been proposed to match the motion of the user with areas of the building based on a given structural map of the environment using graph matching methods [7]–[9].

In [7], authors propose a system to automatically generate the fingerprint-location dataset for an image-based localization system. More precisely, a trainer normally walks in the indoor environment while collecting a video using a consumer grade handheld recording device. The environment is modelled as a piece-wise linear graph and it is assumed that the trainer walks on this graph starting from a certain point. The frames (fingerprints) are labelled for location based on the above-mentioned assumption using particle filters. In [8], [9], authors propose a method than can leverage crowdsourced data to generate fingerprint-location dataset. More, precisely, they process the motion time series from accelerometer and gyroscope to understand the motion patterns of a user. They also use a graph-based representation of the indoor environment. The processed motion data leads to several motion trajectories. The authors then propose methods to attach as many of these motion trajectories as possible together to build a so-called data graph. The resulting data graph is then matched to graph representation of the building using graph matching

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

approaches such as the Hungarian algorithm. These methods show good performance specifically when it is possible to attach many estimated trajectories correctly. In other words, they will not perform as good when the data graph is not covering the building graph, since the Hungarian matching performs poorly for partial matching.

In this work, we focus on the problem of associating motion patterns of a user with her location, and provide an algorithm for *Localization of Motion* referred to as *LocaMotion*. Unlike the previous graph-matching approaches [8], [9], LocaMotion associates certain motion patterns to specific locations of the building using supervised-learning and can handle, in many cases, motion information from a single user walking only in a portion of the building. Unlike [7], Loca-Motion, assumes the motion information is crowdsourced data and relaxes the assumption of a trainer walking in the indoor area. We will elaborate on the Loca-Motion pipeline in following sections.

II. Problem Statement

The question answered by the LocaMotion can be summarized as: given the motion and WiFi measurements from ordinary users in an indoor area, can we associate those measurements to certain parts of the building? The result would be a set of location-labelled WiFi fingerprints obtained solely by the information from the crowd of ordinary users. LocaMotion uses a perspective different from the previous works that will fix some of the shortcomings of the previous works in terms of accuracy and complexity. LocaMotion uses an abstraction of the floor structure referred to as the *floor grid* or simply *grid*. A grid is the collection of all possible pathways on the floor and comprises straight lines connected at corner points. A sample grid on top of the floor plan is shown in Fig. 1. It is assumed that users walk on this grid over the lines and from one point to the neighboring points. A floor grid effectively encapsulates the constraints for human walk in an indoor area. A person residing at a point on a grid can move one *step* further to a random neighboring point on the grid. A sequence of such steps on the grid will generate a *walk*. Suppose the i-th point on the grid, p_i, has the position $\mathbf{l}_i$ represented in a certain coordinate frame chosen to represent the grid. A walk $\mathbf{y}_{1\times L}$ of size L is basically an array of L steps taken on the grid e.g. $\mathbf{y} = [p_2, p_5, p_9, p_3, \cdots]$. If point p_i is at position $\mathbf{l}_i$, then corresponding to $\mathbf{y}$, we can define the array of visited positions $\mathbf{P}_{2\times L}$ comprising the positions of the visited grid points. For example, if $\mathbf{y} = [p_2, p_5, p_9, p_3, \cdots]$, then $\mathbf{P} = [\mathbf{l}_2, \mathbf{l}_5, \mathbf{l}_9, \mathbf{l}_3, \cdots]$ is the matrix comprising the visited positions stacked as columns of $\mathbf{P}$. Suppose a trainer walks $\mathbf{y}$ on a designed grid in an environment. Corresponding to this walk, LocaMotion extracts a so-called *motion feature* $\mathbf{X}$. We will define the proposed motion feature shortly. The pair comprising the motion feature extracted from this walk, $\mathbf{X}$, and the corresponding sequence of points visited, $\mathbf{y}$, constitute a training example $\{\mathbf{X}, \mathbf{y}\}$ for our supervised learning problem. LocaMotion casts the problem of localizing the motion of the user as: *given a certain motion data with motion feature* $\mathbf{X}$ *extracted from a walk by a user, can we localize the path the user took in the environment (indoor localization) or equivalently, predict the label of* $\mathbf{X}$*(motion localization) given that we have sufficient number of training examples i.e., set of walks labelled by the location of visited grid points?* More precisely, LocaMotion trains a classification model that predicts the value of $\mathbf{y}$ given the feature $\mathbf{X}$ extracted from user-sent data. Fig. 2 illustrates how LocaMotion should perform on crowd-sourced data to estimate the locations $\mathbf{y}$ corresponding to motion features $\mathbf{X}$ extracted from a user-sent motion data and associate them with the unlabelled WRSS fingerprints and store them in the dataset of WRSS-location pairs for future WRSS-based localizations.

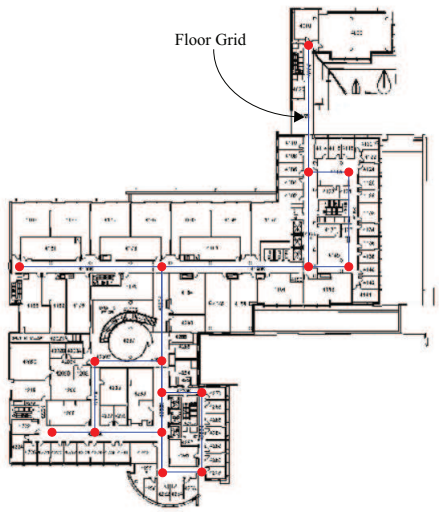

Fig. 1: Floor grid designed for the Bahen Building (UofT).

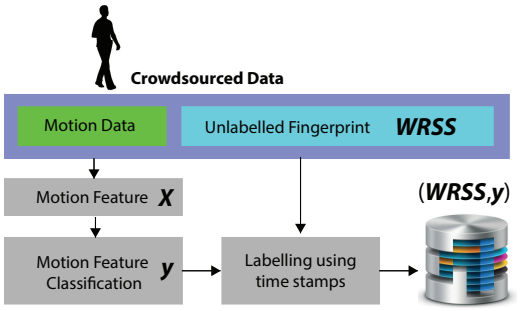

Fig. 2: LocaMotion Overview

If many training examples are collected from more walks on the same grid, the result is a set of N training examples $\{\mathbf{X}^{(t)}, \mathbf{y}^{(t)}\}_{t=1}^{N}$ that can be used to train the classifier. To be able to train the model, one has to perform many random walks to provide the LocaMotion with sufficient number of training examples, which is practically impossible. However, producing simulated random walks is easy and can be used to train LocaMotion. In other words, LocaMotion first generates a large dataset of random walks on a given grid, extracts the motion features and uses these features with known labels (sequences of visited grid points) as training samples. Intuitively, LocaMotion implements virtual users to perform

enough number of walks on the grid, and then models their relative motion behaviour to predict the visited grid points.

III. Motion Features

Let a_{ij}, $\mathbf{a}_{i:}$ and $\mathbf{a}_{:j}$ denote the ij-th element, the i-th row and the j-th column of a matrix $\mathbf{A}$, respectively. The so-called *motion feature*, $\mathbf{X}_{2\times(L-1)}$, is a matrix that represents the consecutive relative motions on the random walk characterized by $\mathbf{y}$ or $\mathbf{P}$ as in previous section. More precisely,

$$\mathbf{X}_{:j} = \mathbf{P}_{:(j+1)} - \mathbf{P}_{:j}\ ,\ j = 1, \cdots L-1. \tag{1}$$

Note that $\mathbf{X}$ captures only the relative motions occurred during the walk regardless of the absolute locations of the points, $\mathbf{P}$, on the walk. Thus, the defined motion feature is *translation-invariant*. Being translation-invariant is important since the trained model will be used to localize future motion patterns and those measurements could be in an arbitrary coordinate system having a random translation with respect to the coordinate frame of the floor grid.

The coordinate system used to measure future motion patterns can also have a shift in rotation with respect to the one used for the grid. To be robust to shift in rotation, LocaMotion includes rotated versions of virtual random walks in the dataset as well. More precisely, for a training example $\{\mathbf{X}, \mathbf{y}\}$ obtained from a simulated random walk, we also include its rotated versions by θ_j in the dataset as well. For every $\{\mathbf{X}, \mathbf{y}\}$, we put $\{\mathbf{R}(\theta_j)\mathbf{X}, \mathbf{y}\}$ in the dataset for all j where $\mathbf{R}(\theta_j)$ is the rotation matrix corresponding to the rotation angle θ_j. The θ_j is systematically sampled from $[0, 2\pi]$ as,

$$\forall j = 1, \cdots, M:\ \theta_j = \theta_0 + \frac{2\pi j}{M}\ ,\ \ \theta_0 \sim U[0, \frac{2\pi}{M}] \tag{2}$$

Therefore, for every random walk sample, one uniform sampling is done according to (2) to generate M rotated versions of the random walk in the dataset. Finally, suppose we have generated the sample $\{\mathbf{X}, \mathbf{y}\}$ using a simulated walk of length L where $\mathbf{y} = \{y_1 y_2 \cdots y_{L+1}\}$. Then, we also add the reverse walk in the dataset as well. The *reverse walk*, $\mathbf{y}'$, is generated from the original walk $\mathbf{y}$, if the user had travelled the visited nodes in reverse order i.e. $\mathbf{y}' = \{y_{L+1} \cdots y_2 y_1\}$. Therefore, for each $\{\mathbf{X}, \mathbf{y}\}$ in the dataset, $\{-\mathbf{X}, \mathbf{y}'\}$ is also added.

Rotation and translation invariances are key aspects of the proposed motion features. Without such characteristics, the correlation between a random walk sample in the dataset and a queried motion pattern could not be measured as these walks are represented in coordinate frames with arbitrary relative translation and rotation with respect o each other.

IV. Dataset Generation (Model Training)

LocaMotion parameters are summarized in Table I. To train the classifier, a walk length has to be chosen to create the dataset. To choose reasonable values one should consider two extremes. On one hand, a very short walk may be very hard to localize as it can happen in many different areas in the building. On the other hand, a walk that is too long may be hard to classify if it is too noisy (error usually accumulates in motion estimation systems). In practice, a system can be trained using several values of L. Then, the queried motion features can be truncated to the nearest L for which LocaMotion is trained (this is our approach in this section). Alternatively, one can first determine L from the queried motion feature and then train using that L (we use latter). Grid Spacing S, is the size of the virtual random step. Smaller grid spacing can be used for finer localization with the cost of higher time-complexity. Number of sampled walks N should be large enough to capture as many motion patterns as possible in different areas in the building so in general a larger dataset means better performance. Two possible limiting factors could be memory budget and time complexity of online localization phase. Note that training is usually done off-line, which makes even large values of N affordable. Number of sampled rotations M, follows a similar discussion to the number of walks. In this paper M is set to 8. At each point on the floor grid, the probability of moving to each neighboring point can be defined based on a motion model. For example, people tend to move forward or turn left or right, more than going back to where they were one step ago. A probability mass can be assigned to each neighboring point to the current location of the user. Sampling from this distribution will determine which neighboring point is the next location.

TABLE I: List of parameters

Name	Description	Typical Values
L	Walk Length (number of steps)	$\frac{1}{3}$ to 1 of grid size
S	Step Size (Spacing)	1 to 10 meters
N	Number of Sampled Walks	set by learning curves
M	Number of Sampled Rotations	8 to 16
α	Probability of going back	1 to 5 %

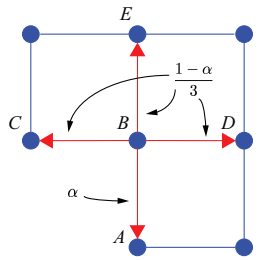

Fig. 3: Portion of a sample grid.

For example in Fig. 3, a user walks from grid point *A* to grid point *B*. Now residing at point *B*, a random step can take the person back to point *A* or either of the points *C*, *D* or *E*. Usually people tend to go forward rather than going back to the previously visited position. In fact this a simplified version of zero-acceleration motion model commonly studied and used in the literature [10], [11]. In Fig. 3, α shows the probability of going back to the last visited point *A*. Also probabilities of visiting the other three neighbors of *B* other than *A* i.e. *C*, *D* and *E* are assumed to be the same. This model for the random step is used by LocaMotion to generate

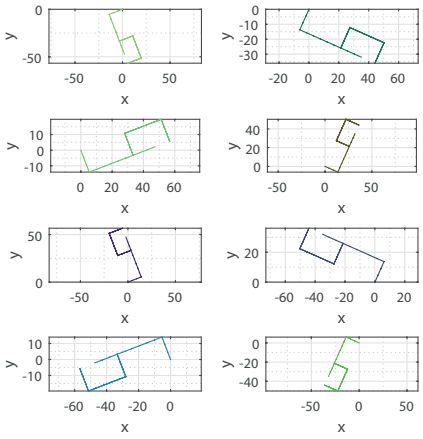

Fig. 4: $M = 4$ training motion features obtained on Fig. 1.

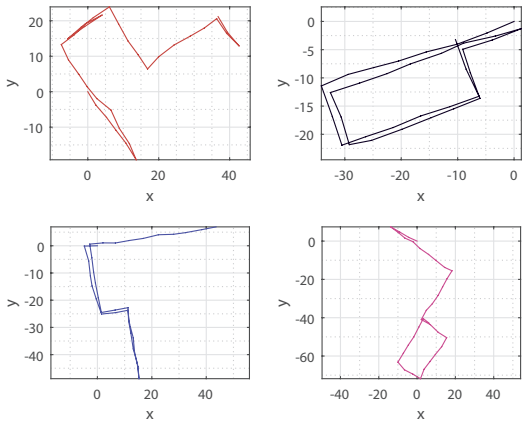

Fig. 5: Sample query motion features to test LocaMotion.

random walks and is intuitively close to human behaviour indoors. The reason LocaMotion uses this model is that it is simple, fast to implement and realistic enough but one can use any motion model here without any limitations to the rest of discussions or proposed algorithms in this paper as long as the model sufficiently complies with real human motion behaviour indoors. The parameter α is the going-back probability and is a design parameter. We use $\alpha = 5\%$ in our experiment. Fig. 4 illustrates a sample motion feature generated on the grid of Fig. 1 for $M = 4$.

V. Localization and Error Metrics

Suppose a query motion feature, $\mathbf{X}^q$, has been extracted from the user-sent information. The classification rule h will predict the label for this feature as $\mathbf{y}^{t^*} = h(\mathbf{X}^q)$. To find t^*, l_2-norm between $\mathbf{X}^q$ and all $\{\mathbf{X}^t\}$ in the dataset is computed and the label of the sample with minimum distance is chosen. The prediction for $\mathbf{X}^q$ using the model h is $\mathbf{y}^{t^*}$ where,

$$t^* = \underset{t \in \text{dataset}}{\arg\min} \left\{ \sum_{i,j} \left(\mathbf{X}^q_{ij} - \mathbf{X}^t_{ij} \right)^2 \right\} \tag{3}$$

This classification rule is basically the nearest neighbor classifier and to find t^*, $O(N)$ feature comparisons have to be made. In practice, to structure the search space and reduce the time complexity of the algorithm, we construct a k-dimensional tree ($k-d$ tree) model of all N training examples. This means, given a query, if the training examples are structured as a L-dimensional tree (L-step walks), we will need $O(\log N)$ comparisons to end up with the most similar training example at the bottom of the tree. We use the canonical implementation of $k-d$ tree in [12], [13]. Note that labels in our application are sequences of grid points. To define the classification error, suppose that the model prediction is $\mathbf{y}' = h(\mathbf{X}^q)$, and the actual label is $\mathbf{y}^q$, then the classification accuracy, CA, is,

$$CA = \frac{1}{L} \sum_i I[\mathbf{y}'_i = \mathbf{y}^q_i] \tag{4}$$

where $I(\cdot)$ is the indicator function taking the value of 1 or 0 if the input argument is true or false, respectively. Basically, CA for a localization task shows what portion of the points on the path where correctly classified (localized correctly). When each node on a path is localized, some error (in metric) happens, which is a measure for localization error. More precisely, in (4), the geographical distance between points $\mathbf{y}^q_i$ and $\mathbf{y}'_i$ is the localization error.

VI. Experiments

A cross validation is used in LocaMotion to set the parameters to reasonable values and get an estimation of the generalization power of the model for future queries. We perform hold-out cross-validations i.e., in each step of the cross-validations, a random walk is generated. The previously built dataset, as a training data, is used to localize the test sample as the query. The error is measured and the experiment is repeated as many times as required to get an estimation of the error distribution. After generating a test random walk for a cross-validation, we also add noise and random rotation to the motion feature $\mathbf{X}^q$ before performing the classification. The added noise is basically a zero-mean Gaussian noise with variance σ^2. The rotation is also randomly selected from a uniform distribution in $[0, 2\pi]$. This is also to make the cross-validations closer to reality in which the queried motion feature has random rotation with respect to dataset and could be noisy due to error in the dead reckoning system. Therefore in (3), $\mathbf{X}^q$ is,

$$\mathbf{X}^q \leftarrow \mathbf{R}(\theta)\mathbf{X}^q + \mathbf{n} \ , \ \theta \sim [0, 2\pi] \ , \ \mathbf{n}_{ij} \sim \mathcal{N}(0, \sigma^2). \tag{5}$$

Fig. 5 illustrates some noisy motion features used as test queries. As discussed in the previous section, it is important to provide a reasonable set of parameters when using LocaMotion. One of the benefits of validation experiments is that we can study the learning curve for the LocaMotion to find a reasonable value for the size of the dataset which is proportional to N. Fig. 6 depicts the learning curve of

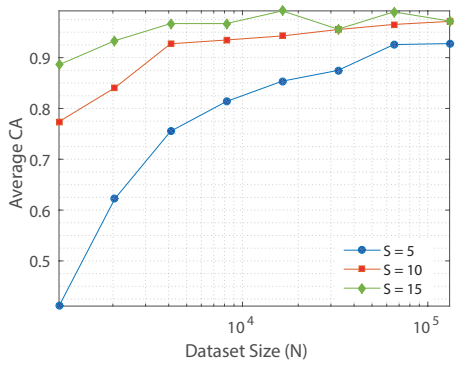

Fig. 6: The learning curve (*ACA* versus N) for 100 cross validations with $S = 5, 10$ and 15 ($L = 34, 15$ and 9) . Noise level is fixed to $\sigma = 1$ meter and $M = 8$.

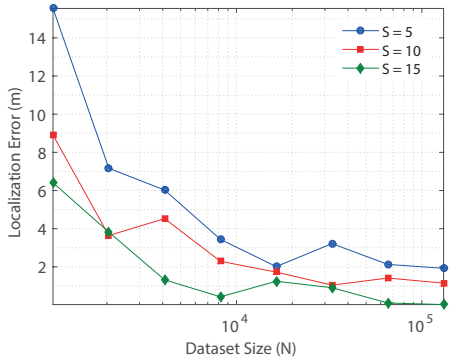

Fig. 7: The learning curve (Localization error versus N) for 100 cross validations with $S = 5, 10$ and 15 ($L = 34, 15$ and 9) . Noise level is fixed to $\sigma = 1$ meter and $M = 8$.

LocaMotion for different values of spacing S, over 100 cross validations as we increase the N. As shown, increasing the size of the dataset increases the Average CA (*ACA*) up to a certain value of N. For example for the case of $S = 5$, increasing N up to 2^{16} samples, increases the *ACA* up to around 93%. As the Learning curve gets near flat, further increasing the N may not be beneficial, hence a reasonable value of N is found. As mentioned before, besides *ACA*, localization error is an important metric to investigate. The final goal of LocaMotion for motion classification is to perform indoor localization. Fig. 7, illustrates the learning curve for localization error for different spacings. As can be seen, the general trend for each value of spacing is decreasing with higher training samples which is in accord with Fig. 6.

One important point regarding both Fig. 6 and Fig. 7 is the choice of spacing. It seems that both *ACA* and localization error show better performance for higher values of S. However, one should note that smaller S provides closer points on the grid, which means higher precision localization (finer localization grid) for future queried motion features. For example, $S = 15$ meters provides better test evaluations if the test queries are generated as in (5). However, if a queried path starts somewhere in between two points with $S = 15$ meters separation, there is a high chance of poor localization due to large spacing and none of the points on the queried path getting matched to any of the grid points. On the other hand, this problem is less severe for $S = 5$ meters due to a much finer grid.

VII. Test Validations

We also performed some real experiments in the Bahen Building at Universoty of Toronto (UofT) (Floor grid of Fig. 3). In these experiments a user takes a walk in the environment while the time series from the phone accelerometer and gyroscope are recorded. LocaMotion first runs an odometer on these time series to generate the user trajectory and then extracts the corresponding motion features as in (1). We use a simplified version of odometers in [14]–[16]. Next, LocaMotion classifies the query motion features to the most similar motion patterns in the dataset and consequently localizes the trajectories. Fig. 8(a) shows a motion feature extracted from a user-sent data in the Bahen Building. The resulting sequence of nodes from classification is shown in Fig. 8(b) i.e. the sequence of grid points [7, 25, 24, 6, 26, 8, 9, 10, 11, 12, 8] which is pretty accurate. In fact the user walked the same path and the classification accuracy is 100%. Clearly, the estimated motion is affected by the drift from the odometer, as can be seen in Fig. 8(a), however, classification shows robustness to this amount of distortion in the extracted motion feature.

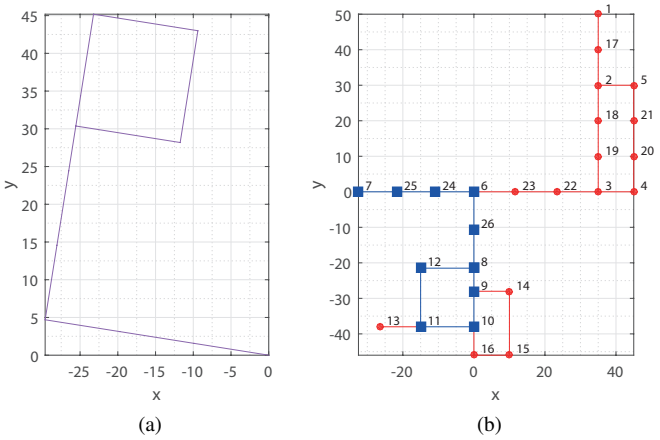

Fig. 8: Using LocaMotion to (a) convert a user-sent trace to a motion feature and then (b) classify it to the corresponding parts of the building.

Fig. 9 illustrates another experiment similar to Fig. 8. The sequence of predicted points as shown is [7, 25, 24, 6, 26, 8, 9, 10, 11]. The ground truth path was [7, 25, 24, 6, 26, 8, 9, 10, 11, 13]. It can be seen that predicted path is reasonably accurate but unlike the previous example is slightly erroneous at the end of path, missing the grid point 13. To further improve the results shown in Fig. 9, one can perform cross

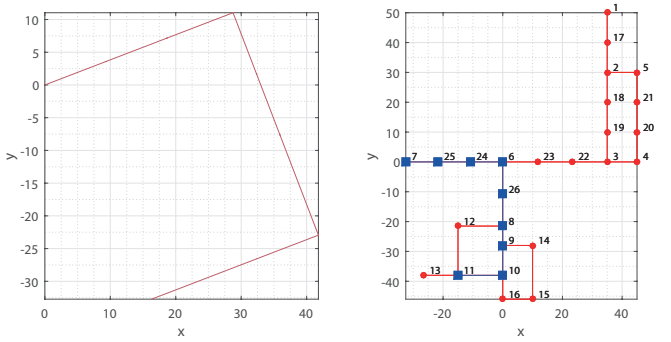

Fig. 9: Using LocaMotion to convert a user-sent trace to a motion feature and then classify it to the corresponding parts of the building.

validations with smaller S values for a finer localization. As we discussed before, smaller S will results in larger L for training, which requires a larger amount of time for a reasonable cross validation accuracy. We have used $S = 5$ for the same motion feature in Fig. 9(a). The result is shown in Fig. 10 showing perfect localization and addressing the issue of Fig. 9(b).

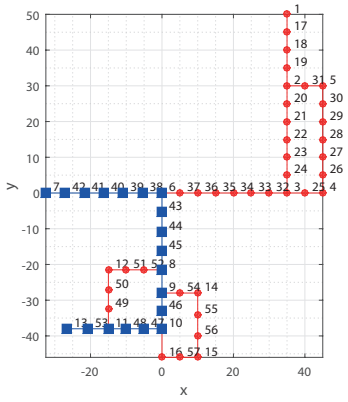

Fig. 10: Using Loca-Motion to convert a user-sent trace to a motion feature and then classify it to the corresponding parts of the building ($S = 5$).

In addition to above use cases for LocaMotion, LocaMotion can be used in conjunction with a WiFi-based localization engine. If WiFi-scans are shared along with motion information by the crowd, LocaMotion can associate WiFi features (WRSS) to locations after localizing the motion features. In other words, LocaMotion can be used to build or update the dataset for a crowdsourcing-based WiFi indoor localization system. Shifting the burden of the training phase in WiFi-based localization systems to the crowd can greatly extend the widespread use of WiFi-based localization systems. To illustrate this capability, we have implemented the scenario explained above. That is, in all previous experiments of LocaMotion, we have also recorded the WRSS time series which is the sequence of scanned WRSS in time during the user's walk. Therefore after localizing the motion features, we can also localize the WRSS values and create the database of location-tagged WiFi fingerprints.

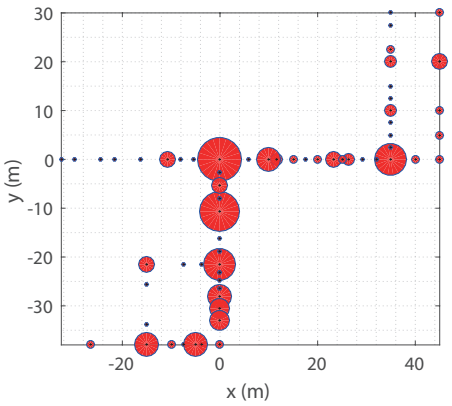

Fig. 11: Geographical distribution of the radio-map generated by LocaMotion using user-sent traces (total of 5 traces).

We performed this process using 5 different time series including those in Fig. 8 and 9, and generated the crowdsourced WRSS-location dataset also known as the *radio-map*. The geographical distribution of the resulting radio-map is shown in Fig. 11. Fig. 11 shows the locations on the grid for which a WRSS is assigned. A larger circle on a grid reflects that more WRSS are assigned to that grid point. We perform a validation test for many different WRSS-locations in the dataset to get an estimation of the localization error using the built WRSS-location dataset. In each validation, a sample in the generated dataset is hold out as query and the rest is used as the dataset and the resulting WiFi localization error is recorded.

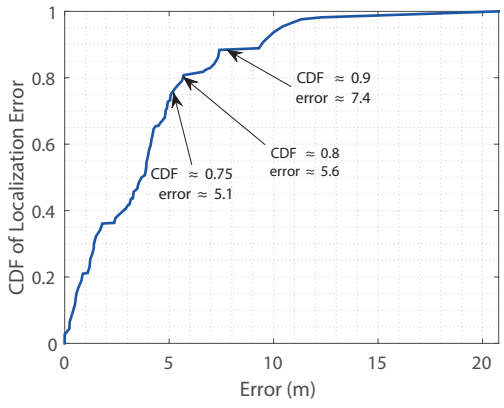

Fig. 12: Cumulative distribution function of the localization error for the radio-map of Fig. 11. The mean localization error is 4.09 meters.

The localization error distribution for the built radio-map is shown in Fig. 12. The mean localization error is about 4 meters. This is an acceptable localization accuracy based on

the fact that average grid spacing S was set to five meters for most of the experiments and we have used only 5 traces.

TABLE II: Comparison between LocaMotion and [8]

	Shahidi et al [8]	**LocaMotion**
Methodology	Graph Matching Hungarian/Stable/K-Best (Deterministic)	Supervised Learning kNN via k-d tree (Probabilistic)
Feature Type	Vector of Shortest Paths	Motion Features
Accuracy	70-95 %	92-98 %
Partial Matching	No	Yes
Preprocessing (offline)	• Floor Graph • Shortest Paths (of floor graph) –	• Floor Graph • Virtual Walk • Train/Crossvalidate • ($k-d$ tree)
Processing (online)	• User Trajectories • Data Graph • Feature Extraction (of data graph) • Matching (Assignment)	• User Trajectories • Feature Extraction (motion features) • $k-d$ Tree Search –
Complexity (online)	$\approx O(n^3)$	$O(\log n)$

Finally, Table II, compares LocaMotion with the mothod in [8] from different perspectives. The last row in Table II, shows the dominant term for the complexity of each algorithm when handling a query. For each algorithm, n is the size of the problem which we explain more precisely. Assume the data graph in [8] comprises V nodes and E edges. [8] have to extract all-pairs shortest paths of the data graph. This process is $O((V+E)\log V)$ for a single node. Repeating for all V nodes leads to a complexity of $O(V((V+E)\log V))$. Usually data graphs are very close to trees i.e., $V=\theta(E)$. Therefore, one can approximate this complexity to $O(V^2\log V)$. The Hungarian algorithm afterwards is $O(V^2E)$. Assuming graphs are close to trees, the dominant complexity would be $O(n^3)$ where we used $V=O(n)$. On the other hand, LocaMotion has to search the $k-d$ tree for the most similar training example to the query. Suppose query is of size L steps. Then this takes $O(\log L)$. Note that, L would have been the size of the data graph in the context of [8] i.e., $L=\theta(V)=O(n)$. Therefore, LocaMotion handles the query in $O(\log n)$ as shown in the Table.

VIII. CONCLUSION

LocaMotion is a system for localization of motion patterns. It is a learning-based algorithm that uses a realistic human motion model and the concept of a floor grid to generate many training samples of random walks on that grid, extracts the so-called motion features and associates them with the corresponding sequence of labels. Consequently, each training example consists of a motion feature associated with a sequence of grid points as a label for that feature. This training dataset is then used to localize future motion features obtained from the user-sent motion data. This way, LocaMotion is a supervised-learning approach that localizes motion patterns for indoor localization. One major application of Loca-Motion, other than being used as a standalone localization engine, is to be used as an unsupervised WiFi-based localization method that generates or updates the database of WRSS-location needed for a WiFi-based localization. In such an application, LocaMotion associates the motion patterns to certain grid points in the building along with the scanned WRSS, hence generating/updating the WRSS-location dataset using only user crowdsourced data. This application of LocaMotion in WiFi localization can greatly reduce the burden of the training/updating the radiomap by leveraging the crowd. This can in turn facilitate the widespread use of WiFi-based indoor localization methods.

REFERENCES

[1] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 6, pp. 1067–1080, November 2007.

[2] S. Adler, S. Schmitt, K. Wolter, and M. Kyas, "A survey of experimental evaluation in indoor localization research," in *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*, Oct 2015, pp. 1–10.

[3] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *Proceedings of IEEE INFOCOM 2000*, March 2000, pp. 775–784.

[4] S. Sorour, Y. Lostanlen, and S. Valaee, "Rss based indoor localization with limited deployment load," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, December 2012, pp. 303–308.

[5] S. Sorour, Y. Lostanlen, S. Valaee, and K. Majeed, "Joint indoor localization and radio map construction with limited deployment load," *Mobile Computing, IEEE Transactions on*, vol. 14, no. 5, pp. 1031–1043, May 2015.

[6] C. Feng, W. Au, S. Valaee, and Z. Tan, "Received-signal-strength-based indoor positioning using compressive sensing," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 12, pp. 1983–1993, Dec 2012.

[7] F. Vedadi and S. Valaee, "Automatic visual fingerprinting for indoor image-based localization applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. PP, no. 99, pp. 1–13, 2017.

[8] S. Shahidi and S. Valaee, "Graph matching for crowdsourced data in mobile sensor networks," in *2014 IEEE 15th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, June 2014, pp. 414–418.

[9] ——, "Hidden markov model based graph matching for calibration of localization maps," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 4606–4611.

[10] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People tracking with human motion predictions from social forces," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 464–469.

[11] S. Kaiser, M. Khider, and P. Robertson, "A human motion model based on maps for navigation systems," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, p. 60, Aug 2011.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[14] A. Mikov, A. Moschevikin, A. Fedorov, and A. Sikora, "A localization system using inertial measurement units from wireless commercial hand-held devices," in *International Conference on Indoor Positioning and Indoor Navigation*, Oct 2013, pp. 1–7.

[15] K. Hili and A. Sammut, "Pedestrian tracking through inertial measurements," in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Oct 2016, pp. 1–8.

[16] R. Zhou, "Pedestrian dead reckoning on smartphones with varying walking speed," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.