Lightweight Indoor Localization System

Mihai Bâce
ETH Zurich, Switzerland
Email: mihai.bace@inf.ethz.ch

Yvonne Anne Pignolet
ABB Corporate Research, Switzerland
Email: yvonne-anne.pignolet@ch.abb.com

***Abstract*— Indoor localization is an important topic for context aware applications. In particular, many applications for wireless devices can benefit from knowing the location of a user. Despite the huge effort from the research community to solve the localization problem, there is no widely accepted solution for localization in an indoor environment. In this paper we focus on constrained devices and propose an extremely lightweight indoor localization system that can be scaled to different devices, from smartphones to smart glasses and other devices.**

We devise a simple yet effective WiFi-based system with low computational complexity, which does not need any additional special infrastructure nor map or an internet connection. Our system relies on IEEE 802.11 Received Signal Strength Indicator (RSSI) values and a dead reckoning module to collect walking trajectories which are further clustered and compressed to build a sensor map. The key novelty of our work is a merging algorithm that can fuse multiple sensor maps.

We evaluate our system in a real world scenario and we show that using the map produced by our merging algorithm we achieve room-level accuracy. Our system is also comparable to state of the art systems, despite the lightweight approach.

I. Introduction

Nowadays, context aware applications are becoming more and more popular and they allow businesses to develop a line of products to target specific customers and needs. A few years ago, there were no localization systems until GPS became widely available. This unveiled many new ways in which companies could interact with their customers and offer extremely valuable services like turn-by-turn navigation. GPS systems have evolved a lot and today's technologies are very refined, achieving centimeter level accuracy. While this is impressive, once a user steps inside, a localization system based on GPS will not work due to the attenuation of the signal strength and other factors that cause the errors to be tens or hundreds of meters.

Recently, there has been a lot of effort to develop an indoor localization systems which can be easily deployed and used in any type of indoor space. Such systems can be split into two important categories: infrastructure-based and infrastructure-less. While one might think that having additional infrastructure improves the localization error significantly, a recent localization competition, the Microsoft Indoor Localization Competition - IPSN 2014 [10], has shown that in practice, having the same conditions and time to calibrate such systems, they behave very similarly. More precisely, infrastructure-less systems achieve a localization error in the same range as infrastructure-based systems. Another important conclusion from the localization competition is that the indoor location problem is still unsolved. It is unclear if this issue is caused by the general methods or techniques or by the poor precision of various sensors.

What we can notice in most of the localization systems available is that most of the results are incremental (e.g. localization error improved only marginally). Thus, instead of focusing on the absolute error, we focus on providing a localization system that requires no prior calibration or maps, no user interaction and low computation complexity.

For consumer related indoor localization applications, an error range of 2-5 meters is sufficient since this provides room level accuracy, on the other hand it is crucial to keep the organisational complexity low [3]. In our paper we present a simple localization system that can be easily extended to different sensors and types of devices and at the same time, maintain a comparable error with state of the art systems. Our focus is to provide a lightweight system with minimal requirements so it can be used for wearables and embedded devices. Apart from the existence of WiFi access points (APs), no specialized infrastructure is needed, and our system works without a map or an internet connection. Its main ingredients are a pedestrian dead reckoning module which offers step detection, orientation and step length estimation, a sensing module to collect WiFi RSSI values, a clustering algorithm to reduce the amount of information to be stored and a merging algorithm to combine several trajectories into a sensor map. While other approaches use rather involved Neural Networks and Particle Filter algorithms, our approach is based on simple algorithms of low computational complexity and thus it can execute all operations on the device itself, no communication with a localization server is needed. On the other hand, if there is a possibility for communication with other devices, the sensor maps can be exchanged and merged easily and thus the potential of crowd-sourcing can be used.

The remainder of this paper is organized as follows. In Section II we review other WiFi-based indoor localization approaches without additional infrastructure. We give an overview of the main components of the system in Section III, followed by a more detailed description in the subsequent sections. Section VIII describes the evaluation of the system in a real-world environment before we conclude the paper in the last section.

II. Related Work

In the past decade a large body of work on Indoor Localization has accrued. It is beyond the scope of this paper to review

978-1-5090-0351-8/15 $31.00 © 2015 IEEE
DOI 10.1109/WMNC.2015.22

all approaches. We focus here on WiFi-based infrastructure-less systems that do not need user interaction nor deploy any specific hardware but identify the location of the device based on WiFi and sensor measurements on the device only. Among these systems most use a localization server that executes the actual localization computations. E.g., the winner of the Microsoft IPSN Indoor Localization Challenge in 2014 [4] lets the server location engine run a nonlinear recursive Bayesian filter to combine incoming location information, then a particle filter method is employed for the estimation of the location. Zou et al. [1] also use a location server where RSSI values are processed in a Single-hidden Layer Feedforward Neural Network architecture. Similarly, Li et al. [7] use augmented particle filtering to fuse signals from WiFi, magnetic field and IMU sensors, requiring offline bootstrapping for a fingerprint database. LiFS [17] builds a high dimensional fingerprint space, preserving distances between fingerprints, with the goal to assign these fingerprints to a physical location with little human intervention.

Another line of research requires a (preprocessed) floorplan, to prevent localization errors by crossing a wall or moving into a non accessible area. E.g., the runner-up of the above mentioned competition [8] combines WiFi RSSI values with Inertial Measurement Unit (IMU) readings. For the localization, sensor fusion with a particle filter is applied, after which a map matching submodule handles and corrects any inaccurate locations. A similar approach is used in [19], by applying linear chain conditional random fields and constraints derived from preprocessed floorplans. Park [14] proposes a HMM-based trajectory matching algorithm to recover user trajectories.

Among approaches that do not require floor plans nor a location server is [20]. This work combines magnetic field and WiFi measurements, using particle filter methods and a similarity Voronoi-graph. It is computationally more demanding than our system. Another approach that tries to create a floor plan using dead reckoning, without any prior information, is PiLoc [9]. The difference from our work is that they cluster walking paths into segments, not points. One of the drawbacks to that approach is that in its current stage, the system can only consider "turns and long straight lines" [9]. On top of this, their system may fail to differentiate intersecting or parallel paths that are not separated by large enough distances. Our approach is not sensitive to such issues since we do not attempt to map the exact floor plan, but rather offer room-level localization accuracy.

In the above mentioned competition, different localization approaches were compared under the same conditions [10]. The average error obtained by infrastructure-less system was within 1.5m and 5.3m.

III. System Overview

In this paper we are proposing a novel indoor localization system for constrained devices. For our proof of concept we use a smartphone as the hardware for our users because it includes a variety of sensors which can directly be used without any complicated calibration efforts. Nowadays, smartphones are a commodity and people have access to them much easier than in the past. This makes it the obvious choice and compromise between accuracy and ease of deployment. The system that we are proposing is based on step detection, WiFi RSSI observations and graph-based methods for representing the environment as a sensor map. Since the computational complexity is low and the sensor map condenses the measured observations, this approach can be applied to other wearable or embedded devices easily.

We apply a Simultaneous Localization and Mapping (SLAM) approach. Doing both localization and mapping at the same time saves us from a separate costly startup phase (whether this implies fingerprinting an area or installing expensive dedicated hardware) and it leads to a continuously improving system. It is worth mentioning that by no site survey we refer to the fact that we do not have a prior data collection phase for our system. Of course by having users walk in the building with their devices we survey the indoor space, but the key difference is that this survey happens already when the users are using the system. Thus, they can already be positioned with regards to a map (if present) or to the paths that they have collected. While we require WiFi access points (APs) to be in place, we do not need an internet connection nor use any additional infrastructure.

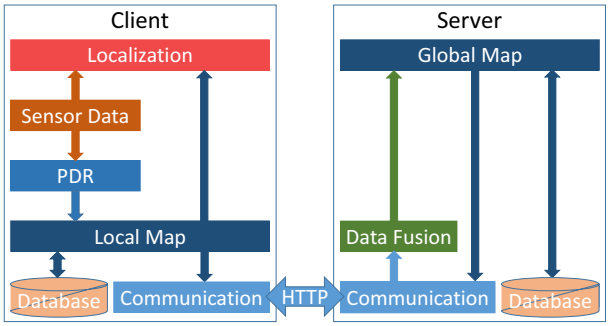

Fig. 1: System overview. The client collects sensor data which is used both as the input to update the local map continuously and for localization. If a connection to the server is available the local map of the client can be merged with the global map constructed by other clients.

Our system can be used as a classical client-server architecture (Figure 1) or running on the client only (left part of Figure 1). In the first case, the client collects sensor data which is used by the Pedestrian Dead Reckoning (PDR) module to detect if a step has been taken (step counter), the length of the step and the orientation. For each step the RSSI values of near-by access points are saved (*Data Collection*). Using this information, the client can update a local sensor map of an area. The local sensor map condenses the available data and only stores a fraction of the measured data (*Clustering and Merging*). The localization algorithm is executed after every step as well and the best matching coordinates of the local sensor map are returned to the user (*Localization*). The local sensor map is only a partial view of the environment which can be fused with a global sensor map to which all clients contribute. To this end, the client and the server communicate

over HTTP and exchange map data. Once the server receives a local sensor map it will fuse it with the current global map (*Merging*). Afterwards, the client will receive an updated map which can be used for localization.

The main responsibility of the server is to fuse multiple local maps into a global map. The localization system that we are proposing, including the merging algorithm, is the key novelty in our work and it is the element that adds the flexibility to extend our solution to difference devices. The only requirements for our system are a WiFi chip and the IMU sensors (accelerometer, gyroscope and magnetometer), which are already present in other devices like Google Glass.

An important aspect of our system is that the server is not always needed. While it supports the crowd-sourcing paradigm, our system can also run independently of the server. A device can build several local maps and fuse them together to create an extended map using the exact same lightweight merging algorithm that the server uses. This gives us the benefit that even when there is no internet connection, our system can still collect data, create a sensor map and use it for indoor localization.

IV. Step-based Data Collection

Using the sensors of a smartphone, the built-in accelerometers can be used as a pedometer and the built-in gyroscope as a compass, to provide heading. This is the most important aspect of a Pedestrian Dead Reckoning module (PDR). While PDR used alone leads to large accumulated errors which are detrimental for localization, it is often used to complement other localization approaches. In our system we use the PDR as a building block to compute coordinates for WiFi measurements. Every time the PDR detects a step, WiFi data is being collected. More precisely, the RSSI values of all APs in the neighborhood are stored. Together with an estimation of the step length and heading the collected data can be assigned to coordinates relative to previous steps.

The step detection algorithm is based on close observation of the stepping process, which is split in two phases: a propulsive phase and a contact phase. In the propulsive phase there is an increase in linear acceleration, while in the contact phase there is a decrease. Just using the norm of the three axis accelerometer measurements, a periodic behavior can be observed, where each period corresponds to a step. The pedestrian dead reckoning module we use is based on [2] for step detection, on [12] for the heading estimation and on [15] for step length estimation. It can be easily replaced by a module applying different approaches to increase the accuracy or to reduce the computational complexity.

A sequence of WiFi measurements and coordinates of steps represents an augmented *walking trajectory*. A walking trajectory is typically defined as a path that a moving object follows through space as a function of time, which also holds true in our case, but with the added difference that we also record the sensor readings from the smartphone for each step.

Using step-based data collection has two advantages. First, data is only recorded when the user is moving which reduces battery power consumption and memory usage. Second, a step as a basic unit for measurements and localization is intuitive to understand for users and developers. On the other hand this approach has drawbacks for areas with moving walkways, as no data would be collected when the user is standing on them and the step length estimation would be misleading if the user is walking. However, both these problems could be overcome with a more refined approach at the expense of more computational overhead (by sampling data even if no steps are detected but storing them only if they differ from previous samples and by adjusting the estimated step length with a factor based on the change rate of data collection). In this paper we focus on scenarios without moving walkways.

V. Clustering

As mentioned in the system overview section, our approach to indoor localization is to collect walking trajectories which are augmented with sensor readings for each step. The main idea that supports our method is based on the observation that once the WiFi signal passes through walls, it suffers from attenuation and the variation is significant enough from one room to the other. Taking into consideration this aspect, it is possible to cluster the collected data. Once the points have been clustered, each sample can be attributed to one cluster. Further on, using this information we can build a connectivity graph between the discovered clusters.

There are many clustering and classification algorithms available, one of the most common, simple and used one is K-Means [11]. The general idea behind the K-Means clustering algorithm is that given a set of N observations, $(x_1, x_2, ..., x_N)$ where each observation is a real d-dimensional vector, it tries to partition the observations into K sets, such as

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x_j \in S_i} ||x_j - \mu_i||^2, \tag{1}$$

where μ_i is the mean of points in S_i and $(K \leq N)$, $S = S_1, S_2, ..., S_k$.

In our case we cluster each walking trajectory as follows. N is the number of steps of the trajectory and x_j is the vector of the RSSI values in dBm measured at step j. If n is the number the AP MAC addresses encountered in the measurements, the i^{th} element of each n-dimensional vector x_j represents the RSSI value of the i^{th} MAC address. For cases when one MAC address is present in one observation, but not the other, we have assumed a default value of -100 dBm. We will explain later which distance metrics we used.

In Figure 2 we can see a walking trajectory that has not been clustered and in Figure 3 we can see the same walking trajectory in which each node belongs to one cluster. Figure 4 shows the connectivity graph between the identified clusters.

For the K-Means clustering, an important step is the way in which we are selecting initial clusters. We have identified two methods that lead to good results. The first one implies choosing the initial cluster centers randomly, making sure that we do not pick the same point twice. The other alternative,

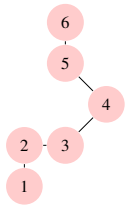

Fig. 2: Walking trajectory before clustering the points

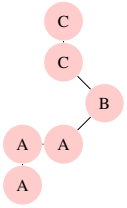

Fig. 3: Walking path after clustering and assigning points

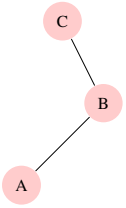

Fig. 4: Connectivity graph between identified clusters

which we believe works better is to choose the initial cluster centers as far away from each other as possible.

Clustering is executed whenever the user stops walking. Thus the number of steps of a trajectory can vary. Hence the number of cluster centers (centroids) varies as well. To achieve room-level accuracy we start the clustering algorithm with a default number of cluster centers (equation 2) of one for every 5 steps. The resulting number of cluster centers after the execution of the algorithm then depends on the trajectory.

$$k_{initial} = \frac{noOfStepsInWalkingTrajecotory}{5}. \quad (2)$$

A. Distance metrics

In any clustering algorithm or when trying to compare two elements, a metric must be defined, which takes into account some properties and based on those properties we can conclude how different the two elements are.

Comparing WiFi RSSI values is not a novel idea. Researchers have tried to model RSSI values so that they can better predict if two values are similar or not, but, despite these efforts, in practice it seems that simpler distance metrics yield good results. In dense urban environments (like shopping malls) we can end up having tens of RSSI entries per measurement.

In our work, we investigated multiple different distance metrics. We will only present the ones which worked best in our case.

1. RSS Absolute Difference [18].

$$d_{\text{Absolute}}(x_i, x_j) = \sum_{l=1}^{n} |x_i(l) - x_j(l)|, \quad (3)$$

where x_i represents the observation vector described previously.

2. RSS Squared Difference.

$$d_{\text{Euclidean}}(x_i, x_j) = \sqrt{\sum_{l=1}^{n} (x_i(l) - x_j(l))^2} \quad (4)$$

There are alternative distance measures like the RSS Stacking Difference used in WILL [16], where researchers claim that the relationship between different APs is more stable than comparing absolute values. Another distance measure is a *top k* approach which means selecting the strongest k access points from each sample and only use those for the distance measure.

However, in practice, we have noticed that the RSS Absolute Difference and the RSS Squared Difference lead to very similar results, but better than the RSS Stacking Difference and the top k approach. On top of this, the RSS Stacking Difference is computationally more expensive because we need to compare each AP to all other. For such a comparison, WILL [16] assumes knowing all the APs in advance, but in our case we did not want to use any prior information.

Taking the above into account, we propose to choose the RSS Absolute Difference as it leads to best results and it is the least expensive in terms of computational complexity.

VI. Merging

Once a user has collected a walking trajectory, it is being clustered, obtaining an abstracted sensor map of a path. This sensor map needs then to be fused with the local map stored on the user's device.

The merging process is the key novelty of our work. Our merging algorithm can run both on the devices themselves and on the server. Once on the device itself, when we merge a new walking trajectory which has been clustered with the current local map from the phone. The second merging happens when the phone sends its local map to the server and the server merges it with the global map, thus enhancing the overall view of the environment.

Sensor map merging can only be performed if we know which nodes from one map are similar to the ones in the other map. This can be achieved by using a similarity matrix. For creating a meaningful similarity measure we will use the distance measures that we have explained in the clustering section of this paper. The distance measurement, represented by a number, is transformed into a similarity metric, which represents a value between 0 and 1. 0 similarity means that the nodes are completely different, while a similarity of 1 means that the nodes are identical.

The walking trajectory and the local sensor map are represented as an undirected path graph and a general undirected graph respectively, where each vertex has attributes to represent the WiFi data collected and the coordinates derived from PDR.

Before describing the similarity measure, there are a few important aspects to consider when defining a new way to asses the similarity. As stated by Nikolic [13] some of the most important properties of a similarity measure are the following.

- If a map is compared to itself, each node of the map should be most similar to itself.
- The similarity scores should have a fixed range, the similarity of a node to itself always takes the maximal value.
- A similarity score should be meaningful in itself. Thus, it is impossible to conclude if two nodes are similar or not, but we can conclude if a pair of nodes is more similar than another pair of nodes. This is one of the drawbacks of using a pairwise approach, by building the similarity matrix.

To obtain the properties above we use a similarity measure described by D. Koutra et al. [5], which can be used for any distance metric that we use.

$$s_{i,j} = 1 - \sqrt{\frac{d_{i,j}}{max_{k,l}(d_{k,l})}} \tag{5}$$

where s_{ij} represents the similarity and d_{ij} represents the distance between two nodes, i and j, and each node is a vector of stored measurement values (see x_i). We use the absolute, the euclidean and the top k distance in the evaluation part of this paper. I.e. given a set of positions each with a measurement vector we can use the similarity definition to decide which positions can be merged. In localization, this similarity measure can be used to decide which position is closest to the currently measured values.

A. Similarity Matrix

The similarity matrix is a way in which we can measure the pairwise similarity between the nodes of two maps.

Let's assume we have two graphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$, where V_i denotes the set of vertices and E_i denotes the set of edges for graph i.

The similarity function s is a function defined by M. Nikolic [13] as $s : D_1 \times D_2 \rightarrow R$, where D_1 and D_2 are possibly equal sets. The similarity measure over the nodes of two graphs can be represented by a similarity matrix $S = [s_{ij}]$, the size being determined by the cardinality of the sets representing the vertices, $|V_A| \times |V_B|$. The element s_{ij} denotes how similar node i is to node j.

B. Graph Matching based on the Similarity Matrix

We mentioned in the previous section that once we have the similarity matrix we want to find what is the best matching between the nodes of one graph and the nodes of the other graph. When we start, the first graph represents the clusters of the first walking trajectory, and the second graph the clusters of the second walking trajectory. After the merging, we will have a new graph that represents the information from both trajectories. This graph will then be used for further merging operations. In our case, for the matching, we only look at nodes themselves and not at the edges. The matching problem is a well known problem in engineering, computer science and many other fields and sometimes it can be found under the name of bi-partite graph matching, maximum network flow problem or even the stable marriage problem.

The graph matching problem can be stated formally in the following way. Let's consider two graphs, $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ which we would like to merge. In order to merge these two graphs (which are abstractions of the sensor maps) we need to identify which nodes from G_A correspond to which nodes from G_B.

The first step is to compute the distance matrix D using a distance metric described in the Distance Metrics section. We compute the pairwise distance between the nodes from G_A and the nodes from G_B, where $d_{i,j}$ represents the distance between node i from graph G_A and j from graph G_B. m and n represent the number of vertices in G_A and G_B.

Further on, we take the distance matrix and we compute the similarity matrix S using the metric described previously in this paper. $s_{i,j}$ represents the pairwise similarity from the distance matrix using equation 5.

The similarity matrix can be further transformed into a cost matrix C, where $c_{i,j}$ represents the pairwise cost calculated from the similarity matrix using equation 6. The cost is a value between 0 and 1, where a cost of 0 means that two nodes are identical and a cost of 1 means that the two nodes are different.

$$c_{ij} = 1 - s_{ij}. \tag{6}$$

Now that we have the cost matrix, we can rephrase the graph matching problem in a different way.

Suppose we have n resources which we want to assign to n tasks on a one-to-one basis. Suppose also that we know the cost of assigning a given resource to a given task. We wish to find an optimal assignment, one which minimizes the total cost. Besides this, there is also the unbalanced version of this problem, the one in which we have n resources and m tasks, knowing that $n \neq m$. Both problems can be solved using the Hungarian method [6] and they involve using a cost matrix.

In our work, we have used the Hungarian method for performing the matching between the nodes of two different graphs. In practice, it is very common to have graphs of different sizes, which is slightly different to the original version of the algorithm that we are using. However, the simplest solution to solve this issue and be in the classical case for the Hungarian method, which is a square matrix, is to pad either the columns or the rows. The padding operation only happens after obtaining the cost matrix, otherwise we would have extra rows or columns, which would add overhead and useless calculations to each step. Using the cost matrix we can apply the Hungarian method and get the optimal assignment with a minimum cost and the highest similarity between the nodes. This will be a n to n matching which we can further filter based on the cost/similarity.

An important aspect to consider when finding the matching between two graphs using the described method is that we will always find a matching for each node of the graph containing fewer nodes. Not all these matchings make sense, some of them have very low similarity values. We filter them out using a threshold t. In our system, we have manually set this value to 0.7 based on our experiments.

The WiFi observations of the nodes which match are combined, using the newer value for each AP, but not deleting APs for which only one measurement exists, as it might be temporarily down. Note that the heading information derived from the PDR is absolute. Thus the coordinates of the larger graphs are used as the coordinates for the merged graph. To shift the coordinate values of the nodes without a match, the shift between the best matching nodes is used.

Note that it can make sense to vary the threshold value over time. Depending on the application domain it might be beneficial to start with a relatively low threshold to ensure that more samples populate the map. At a later point in time the threshold can be raised to be more restrictive and save memory. To favor newer samples, the similarity measure can be adjusted or other aspects (like the coordinate distance) can be added to refine the merging process. As our goal was to investigate what accuracy we can obtain with a very simple and lightweight implementation running on constrained devices, we omit such optimizations.

VII. Localization

In the previous sections we have described a way in which we can build sensor maps and merge them to achieve a more detailed representation of the environment. However, when building an indoor localization system we also need to be able to identify our current position with regards to the available map.

For this purpose, we use a very common and simple approach. Every time a step has been detected, a new location query is sent to the localization component. For predicting the best location we use the distance metric that we have previously defined and find the closest cluster center from our condensed map. The coordinates associated with this cluster center are then the proposed coordinates of the current location.

Since the cluster centers are a condensed representation of the area and we use only one of them for the localization the accuracy of this approach is bounded by the number of cluster centers a map contains. As we strive for room level accuracy and weigh low computation complexity more important, this approach fits our requirements very well. Moreover it is easy to exchange this component with a more sophisticated approach using several cluster centers or additional information.

VIII. Experiments

We implemented a prototype of our localization system for the popular Android platform. As test devices we have used Samsung Galaxy S4 mini devices, which have all the necessary sensors: WiFi 802.11 chip, accelerometer, gyroscope and magnetometer.

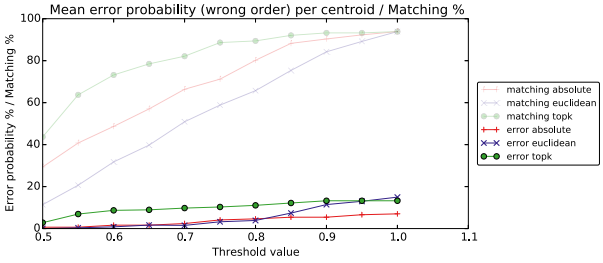

Fig. 5: Matching ratio and error probability for graphs of the same path.

We tested our system in a real-world environment, more specifically in two office buildings, one belonging to a university and one belonging to a company. The test areas cover several offices and long corridors which allowed us to get a good estimate on the performance of our system.

Before describing the experiments, it is important to mention the constraints that we have set. Taking these constraints into account we expected that our results are not as good as systems that use additional infrastructure or rely on a time consuming site survey phase, but we still managed to achieve comparable results.

- We do not use any infrastructure beyond existing APs. No knowledge about the APs is assumed in advance, we do not use a floor plan as a ground truth reference and we do not know the starting point.
- We only use the information that we can get from the smartphone, using the built-in sensors.

A. Merging

To determine the difference between the considered distance metrics and to find a good threshold value t, we performed the following experiments. We collected three separate walking trajectories of ten paths of 10m to 50m length in different buildings. For each pair of trajectories belonging to the same path we computed what percentage of cluster centers was matched with a cluster center from the path (*matching ratio*) and what the probability of picking a cluster center without respecting the order is. While the highest matching ratio is achieved by the topk distance metric, it also features the highest error probability ($< 10\%$), see Figure 5. For paths that do not have any nodes in common, all distance metrics work without invalid matchings up to $t = 0.75$. Considering partly matching paths confirms that a threshold of around $t = 0.75$ is suitable for the environments of our experiments. Since the performance of the different distance metrics varies and there is a tradeoff between matching ratio and error probability, there is no clear favorite. For the sake of simplicity we propose to select the absolute distance metric because it features lower computational complexity than the others.

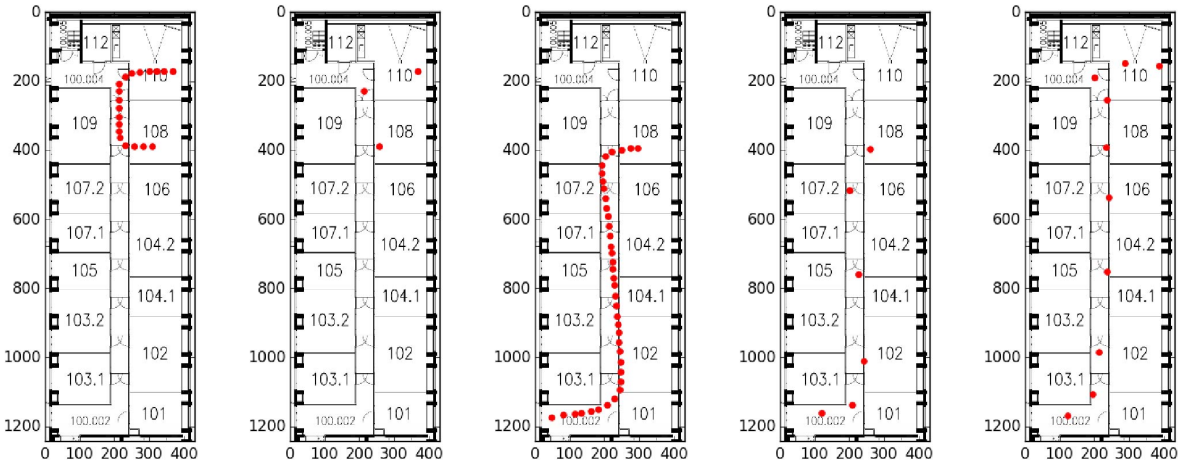

Fig. 6: The sensor map merging process. (a) Walking trajectory 1. (b) Cluster centers for walking trajectory 1. (c) Walking trajectory 2. (d) Cluster centers for walking trajectory 2. (e) Merging multiple paths, including walks 1 and 2.

Figure 6 illustrates the merging process in a part of an office space we used for our experiments. Figures 6a and 6c represent two walking trajectories, which are clustered, the result can be seen in Figures 6b and 6d. The result of matching multiple walking paths, including the two described previously, can be seen in Figure 6e. Note that the map is not necessary for any part of the computation, we just use it here to illustrate our results.

B. *Localization*

For the Localization experiment, we tried to replicate the same requirements as the Microsoft Indoor Localization Competition from IPSN 2014 [10]. The purpose of the competition was to offer a standardized way to evaluate different localization systems in the same environment. While it was impossible to replicate the same environment, we kept the same evaluation scheme.

We selected an origin point that we use as a reference for the coordinate system. All the locations must be reported relative to the origin point as two dimensional coordinates in m (i.e. (1.2m, 2.5m)). We then marked multiple points on the floor of the evaluation area and calculated the X and Y coordinates of these points with respect to the predefined origin.

The test environment can be seen in Figure 7. Figure 7a shows the walking trajectories that we have used for building the sensor map. Figure 7b shows the ground truth location of the test points that we used to measure the localization error. For each point we collected 1200 WiFi measurement samples to be able to compute a meaningful per point localization error. The actual localization experiment then consisted in determining the most similar node of the map for each test point. The

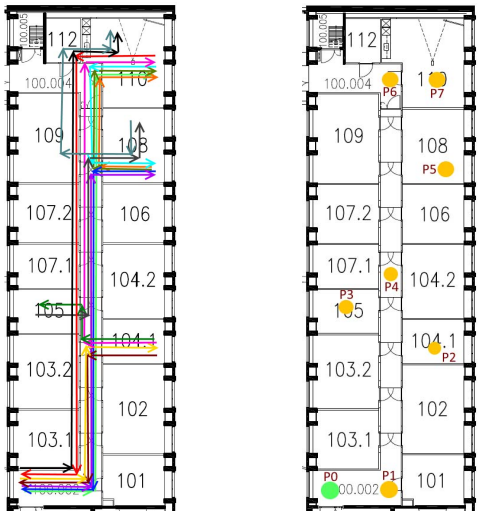

Fig. 7: The localization setup. (a) The 12 walking trajectories used for building the sensor map through our map merging algorithm. (b) The test points used as ground truth for obtaining the localization error.

error in this case is the euclidean distance between the ground truth coordinates of the test points and the coordinates of the most similar node.

Similarly to [9], we compare the average localization error

Approach	Avg. error	Requirements
PiLoc [9]	$1 \sim 3m$	WiFi + IMU
MaWi [20]	$2 \sim 5m$	WiFi + Magnetic
MapUme [4]	$\sim 2m$	WiFi Fingerprinting
Laoudias et al. [8]	$\sim 2m$	WiFi + IMU + Fingerprinting
LiFS [17]	$3 \sim 7m$	WiFi + IMU + Floor plan
Our system	$1 \sim 6m$	WiFi + IMU

TABLE I: Comparison to other localization systems.

to other indoor localization systems. Since implementations of these systems are not publicly available we had to rely on results reported in the literature. What we can notice in Table I is that our system is on par with most of the current state of the art systems. However, in our case no effort is required, no fingerprinting or site survey has to be done in advance. If the user enters an area that is not covered by the current map, the system can determine an estimate of the position based on dead-reckoning only. At the same time the map is being extended for this area for subsequent queries about it without any user interaction. One way to improve the localization error is by improving the dead reckoning module. [21] propose an algorithm called A^3 which greatly improves the tracking error compared to the standard Android code.

IX. Conclusion

In this paper we propose a novel indoor localization system that takes user-generated walking trajectories augmented with sensor readings, clusters them and merges them into a map. The novel merging approach we propose is of low computational complexity and can run both on the client devices or on a server to fuse maps from several devices. Despite using no prior information and without any site survey we manage to achieve a comparable localization error to state of the art systems with more computational complexity, prior information or infrastructure. Having fully implemented our system on real devices and based on real world validation, our localization system is promising and introduces a novel idea in processing crowd-sourced sensor data. Taking into account the large amount of different devices with WiFi and sensors entering the market every year, we believe that our work can be ported to other devices like embedded devices, smart glasses and smart watches which are becoming increasingly popular and powerful.

X. Acknowledgments

We would like to thank Prof. Patrick Thiran and Vincent Etter from École Polytechnique Fédérale de Lausanne for their support and guidance during this project. We would also like to thank Ettore Ferranti from ABB Baden, Switzerland for his useful feedback.

References

[1] H. Zou, H. Jiang, and L. Xie. Wifi based indoor localization system by using weighted path loss and extreme learning machine. *Microsoft IPSN Indoor Localization Competition* (2014).

[2] Kihlberg, J., and Tegelid, S. Map aided indoor positioning. *Master Thesis, Linköping, Sweden* (2012).

[3] Kjærgaard, M. B., Krarup, M. V., Stisen, A., Prentow, T. S., Blunck, H., Grønbæk, K., and Jensen, C. S. Indoor positioning using wi-fi–how well is the problem understood? In *International Conference on Indoor Positioning and Indoor Navigation*, vol. 28 (2013).

[4] Klepal, M., and Beder, C. Mapume–wifi based localization system. *Microsoft IPSN Indoor Localization Competition* (2014).

[5] Koutra, D., Parikh, A., Ramdas, A., and Xiang, J. Algorithms for graph similarity and subgraph matching. https://www.cs.cmu.edu/jingx/docs/DBreport.pdf, 2011. [Online].

[6] Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly 2*, 1-2 (1955), 83–97.

[7] L. Li, C. Zhao, G. Shen, and F. Zhao. Localization with multi-modalities. *Microsoft IPSN Indoor Localization Competition* (2014).

[8] Laoudias et al. Accurate multi-sensor localization on android devices. *Microsoft IPSN Indoor Localization Competition* (2014).

[9] Luo, C., Hong, H., and Chan, M. C. Piloc: A self-calibrating participatory indoor localization system. In *Proc. 13th Symposium on Information Processing in Sensor Networks (IPSN)* (2014).

[10] Lymberopoulos, D., Liu, J., Yang, X., Choudhury, R. R., Sen, S., and Handziski, V. Microsoft indoor localization competition: Experiences and lessons learned. *SIGMOBILE Mob. Comput. Commun. Rev. 18*, 4 (Jan. 2015), 24–31.

[11] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, California, USA (1967), 281–297.

[12] Madgwick, S. O., Harrison, A. J., and Vaidyanathan, R. Estimation of imu and marg orientation using a gradient descent algorithm. In *International Conference on Rehabilitation Robotics (ICORR)* (2011).

[13] Nikolić, M. Measuring similarity of graph nodes by neighbor matching. *Intell. Data Anal. 16*, 6 (Nov. 2012), 865–878.

[14] Park, J.-g. *Indoor localization using place and motion signatures*. PhD thesis, Massachusetts Institute of Technology, 2013.

[15] Renaudin, V., Susi, M., and Lachapelle, G. Step length estimation using handheld inertial sensors. *Sensors 12*, 7 (2012), 8507–8525.

[16] Wu, C., Yang, Z., Liu, Y., and Xi, W. Will: Wireless indoor localization without site survey. *IEEE Trans. Parallel Distrib. Syst. 24*, 4 (2013).

[17] Yang, Z., Wu, C., and Liu, Y. Locating in fingerprint space: Wireless indoor localization with little human intervention. In *Pro. the 18th Conference on Mobile Computing and Networking*, Mobicom (2012).

[18] Yang, Z., Wu, C., and Liu, Y. Locating in fingerprint space: wireless indoor localization with little human intervention. In *MOBICOM* (2012), 269–280.

[19] Z. Xiao, H. Wen, A. Markham, and N. Trigoni. Lightweight map matching for indoor localization using conditional random fields. *Microsoft IPSN Indoor Localization Competition* (2014).

[20] Zhang, C., Luo, J., and Wu, J. A dual-sensor enabled indoor localization system with crowdsensing spot survey. In *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on* (May 2014), 75–82.

[21] Zhou, P., Li, M., and Shen, G. Use it free: Instantly knowing your phone attitude. In *Proc. 20th Conference on Mobile Computing and Networking*, MobiCom (2014).