Reliable Trajectory Classification Using Wi-Fi Signal Strength in Indoor Scenarios

Martin Werner, Lorenz Schauer, Andreas Scharf
Mobile and Distributed Systems Group
Ludwig-Maximilians-University Munich, Munich
martin.werner@ifi.lmu.de, lorenz.schauer@ifi.lmu.de, scharfa@cip.ifi.lmu.de

***Abstract*—The time-series nature of human movement inside buildings can be exploited for common tasks of location-based computing. With this paper, we propose to use Wi-Fi signal strength measurements directly to infer the trajectory in comparison with a database of trajectories removing the need for accurate map information or fingerprint databases. A trajectory consists of a time-series of sensor readings of all Wi-Fi signals in reach measured by a mobile device. Starting from these measurements, we discuss several possibilities of denoising, filtering and classification of trajectories to improve our approch. By using a variant of the Douglas-Peucker algorithm we reduce the amount of computation without severe degradation of classification performance. Furthermore, we increase platform scalability by using a fast filter operation based on the Jaccard index of presence of access points to prune irrelevant trajectories early. With respect to our setting, the Fréchet-distance between trajectories has proven to be a very good choice outperforming dynamic time warping. Finally, we intorduce several data-driven trajectory segmentation schemes in order to be able to match partial trajectories early. The evaluation is based on the collection of trajectories in specific situations including staircases, hallways and movement inside a single room. With this approach, we are able to reliably classify trajectories without an intermediate step of calculating spatial position. This results in increased stability with respect to local changes in the environment, as these changes only affect a small part of a longer trajectory.**

I. Introduction

In the last decade, wireless and mobile computing have reached a level in which high-bandwidth Internet access has become widely available. Moreover, modern smartphones are equipped with a lot of sensors able to capture information about the surroundings of a mobile user. Position determination outside buildings is well-established using two technologies available to the mobile user, namely GPS for position determination and Wi-Fi and cell id proximity estimation over cloud services. However, inside buildings GPS is often unavailable and the achievable accuracy of Wi-Fi and cell id positioning as provided by major Internet companies does not suffice. To close this gap, a lot of research has been done with respect to indoor positioning using existing infrastructure such as Wi-Fi or Bluetooth. In general, there are a lot of techniques that work, but all these techniques have in common that they need a lot of preparation to work involving the generation of detailed signal maps and efforts to keep them up to date. From this research, one can also learn that Wi-Fi signals are usually not used to directly infer a position, rather a pattern matching approach with a fingerprint database has to be employed.

Instead of relying on isolated location measurements, the time series nature of human movement inside buildings can be exploited for some very common tasks of location-based computing. A proactive information service is usually not interested in the current, actual and correct location of a mobile user, but rather in information about the situation of the user such as future location and activities. However, most of these understand trajectories as a time series of spatial location and generate it from a layered structure, where location inference and trajectory-based computations are isolated.

These approaches suffer from the problem that signal anomalies negatively affect location accuracy (e.g., the expected error or mean squared error) and, hence, sophisticated tracking systems enforce linear movement and reduce jumping position estimates. Hence, signal anomalies are suppressed and hidden from higher layers of such layered architectures.

With this paper, we propose to use the signal measurements directly to infer the trajectory in comparison with a database of trajectories removing the need for accurate map information and fingerprint databases and providing a readily deployable system. We propose an indoor Wi-Fi-based trajectory mining scheme in which a trajectory consists of a time series of sensor readings of all Wi-Fi signals measured by a device. Starting from this measurements, we discuss several possibilities to denoise, filter and classify trajectories with a thorough evaluation. The main contributions of this paper are:

- An effective and efficient trajectory mining approach based on sensor data directly and
- a thorough analysis of the choices made with a demonstrative application.

The remainder of this paper is structured as follows. The next Section II reviews related work. Afterwards, some technical background needed to provide trajectory-based services is explained in Section III. Section IV describes our approach and the involved steps. Section V explains the dataset, which was recorded to evaluate the system and Section VI gives evaluation results. Section VII concludes the paper and gives hints on future work.

II. Related Work

In recent years, a lot of research has been done trying to predict the user's next location. Some related work uses trajectory computing approaches based on positioning data provided by GPS [1]–[3]. However, these systems are only applicable outside of buildings and can only be transformed to

the indoor situation using a layered architecture together with some positioning system. Further research is trying to predict the next location according to the actual point in time. These include estimating the location a user will visit within the next ten minutes [4], or the next location checkin within a location-based social network such as *Foursquare* [5]. Some systems solve even simpler problems using less sensor information, such as predicting the time when a person is at home or not [6]. Another view towards predicition is given by finding periodical patterns in location data [7]. It is also possible to use calendar data and create user-specific decision trees, while the prediction is finally based on the actual location, date, time, duration of stay and holiday time [8]. Other research investigate the probability that a user moves from one location to another on various points in time [9]. A more theoretic work ensures a bound of predictability measuring the entropy of trajectories of mobile phone users and give 93% potential predictability. A lot of work has also been done with respect to chosing the right prediction framework including dynamic Bayesian networks, multi-layer perceptrons, Elman networks, Lempel-Ziv, and Markov predictors [10], [11]. In some research, Order-2 Markov predictors have been proposed and evaluated to perform well [12], [13].

III. Technical Background

A. Trajectory Similarity Measures

The primary ingredient in a trajectory-centric classification or clustering framework is a technique, which consistently assigns a distance to pairs of different trajectories. The topological relations between trajectories can become arbitrarily complex, they can be identical, they can intersect, they can be far away from each other, they can have the same shape, etc. For this paper, we describe some notions of distance, which are well-suited for classification of trajectories. However, there exist numerous other notions of trajectory distances, which could not be incorporated in this paper. However, the following distances are typical for several domains and have found wide adoption.

1) Hausdorff-Distance: The Hausdorff distance is defined for all subsets of a metric space. When applied to trajectories or time series data it is usually taken with respect to all points lying on the trajectories. The Hausdorff-metric is based on another metric δ, which is used to calculate the distance between points.

$$\delta_{HD}(A,B) = \max\{ \sup_{a\in \mathrm{Im}(A)} \inf_{b\in \mathrm{Im}(B)} \delta(a,b), \sup_{b\in \mathrm{Im}(B)} \inf_{a\in \mathrm{Im}(A)} \delta(a,b)\}$$

In general, the suprema and infima are difficult to calculate. However, for trajectories represented as polygonal lines, it is actually possible to efficiently calculate the Hausdorff distance by analyzing critical values. Note that the Hausdorff distance is defined on the set of points of a trajectory only and does not reflect the time-domain nature of the sequence. To account for this sequential structure of a track, Fréchet defines another metric which reflects some more of the structure of a trajectory while still being based on an elementary distance between points.

2) Fréchet Distance: The Fréchet distance is best described informally: Imagine a dog and his owner moving on two separate paths. They are not allowed to move backwards. The Fréchet distance is the minimum length of a leash needed to keep the dog and its owner connected. Formally, this can be formulated as taking into account all possible reparametrizations of the trajectories:

$$\delta_F(A,B) = \inf_{\alpha,\beta} \sup_{t\in[0,1]} \delta(A(t),B(t)),$$

where $\alpha,\beta : [0,1] \to [0,1]$ are continuous, non-decreasing, surjective maps from the unit interval into itself called reparametrizations. Again, for polygonal trajectories, efficient algorithms exist, which can be used to calculate the Fréchet distance in $\mathcal{O}((p^2q + q^2p)\log pq)$ time for trajectories of p and q vertices respectively [14]. Allowing some error bounded by the longest segment in the trajectory representation, even a running time of $\mathcal{O}(pq)$ can be achieved [15].

3) Jaccard Metric: A completely different approach is given by the Jaccard Metric, which is not based on an underlying metric δ. Instead, it is based on a cell subdivision of the world in symbolic locations usually represented by labels. Note that one place can have multiple labels, e.g., the visible GSM cell identifiers at a specific location. Using these labels, a trajectory can be naturally assigned the set of labels that are contained in it. For two such sets A and B, the Jaccard index

$$J(A,B) = \frac{|A| \cap |B|}{|A \cup B|},$$

gives a measure of set similarity which takes values in $[0,1]$. Based on this index, the Jaccard metric is obtained by setting

$$\delta_J(A,B) = 1 - J(A,B)$$

There exist very elaborate techniques to quickly calculate the Jaccard index between a set of sets (e.g., a dataset) and a single instance based on sparse matrix multiplication.

4) Dynamic Time Warping: Dynamic time warping (DTW) is a distance definition, which has found wide adoption in the tracking domain. Basically, it compares two trajectories point by point using an underlying metric δ. However, the time between two trajectories is adopted by time warping. Measurements can be used more than once, but all points have to be considered. Formally, this results in the following recursive definition:

$$\delta_{DTW}(a_{1..n}, b_{1..m}) = \delta(a_n, b_m) + \min \begin{cases} \delta_{DTW}(a_{1..n-1}, b_{1..m-1}) \\ \delta_{DTW}(a_{1..n-1}, b_{1..m}) \\ \delta_{DTW}(a_{1..n}, b_{1..m-1}) \end{cases}$$

This distance can be computed in $\mathcal{O}(mn)$ time [16]. Unfortunately, DTW does not provide a metric as it does not fulfill the triangle inequality.

B. Trajectory Preprocessing

Preprocessing of collected raw data is often essential to reduce the processing overhead in subsequent phases, to simplify the classification and finally, to increase the classification performance. For that purpose we have to select the useful data out of the whole dataset in a first step. Furthermore, the

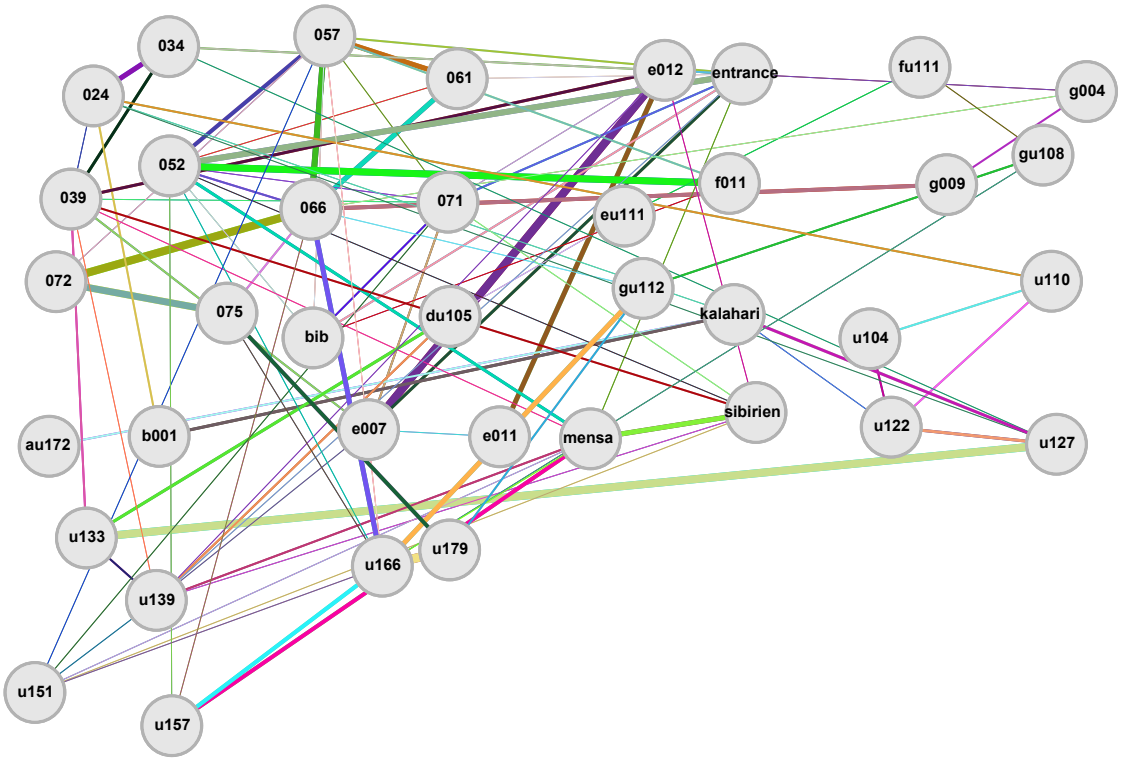

Fig. 1. Dataset represented as a graph of interconnected labels

preselected trajectories can be simplified and smoothed by the following techniques:

1) Douglas Peucker Algorithm: The Douglas Peucker Algorithm [17] serves for compression and smoothing of a trajectory by approximating the original trajectory with less intermediate points. As an initial step of Douglas Peucker, the algorithm approximates the original trajectory by a simple line between the start and endpoint. Then it splits the initial line into two line segments by finding a suitable split point. Such a split point is defined as the sample point of the original trajectory which has the highest perpendicular Eucledian distance to the approximate line segment and, hence, has the highest contribution on the sum of the perpendicular distances of all points in the trajectory. The algorithm continues recursively with both segments until no more split point can be found in the considered segment. This is the case when all sample points of one segment have a perpendicular Eucledian distance smaller than a predefined error threshold ϵ. Thus, the result of Douglas Peucker is an accurately simplified trajectory where all points are within ϵ range to the origin [18]. In computer graphics, ϵ is often chosen in the order of one pixel. Hence, the Douglas-Peucker simplified trajectory looks exactly like the version of the trajectory drawn to a bitmap of pixels.

2) Fast Fourier Transformation Denoising: The fast Fourier transformation (FFT) computes the discrete Fourier transformation (DFT) and its inverse in an efficient way. The DFT is an operation to transform functions to the frequency domain. This technique is used in many fields, where the frequency structure of a signal contains more useful information than the original domain. In order to smooth and denoise collected trajectories, the measured samples are converted into the frequency domain for each dimension and the coefficients of high frequencies are suppressed. The inverse FFT is then used to transform the function back into its original domain and the smoothed trajectories are shown as result.

C. Trajectory Classification

To classify measured time series we use a simple k-next-neighbors (kNN) approach. This technique is a lazy learning algorithm in which an incoming trajectory is compared to all trajectories in the dataset. It takes as input the measured time series and calculates the k next neighbors in the dataset based on a certain distance function. The trajectories are assigned to classes, typically the target points. With a voting approach, we return the class, which contains most of the k next neighbors of the query trajectory. Hence, the measured time series is put into the class with the highest similarity. Depending on the actual service, different conclusions can be drawn from the set of the top k candidates: In proactive information applications, the set of endpoints of these trajectories can be used as an input. For inroute awareness, points of interests near to any of the top k routes can be calculated and scored against a personal profile of interest. In this way, it would even be possible to propose taking a different way towards an estimated goal, which contains more interesting points for the current user. Similar approaches have been made with taxi data based routing. Therefore, mobility traces of taxis were collected and used to route users towards their respective targets. This can also become interesting inside complex buildings, where it might be difficult to find the most efficient way inbetween two locations, especially, if no map information is available.

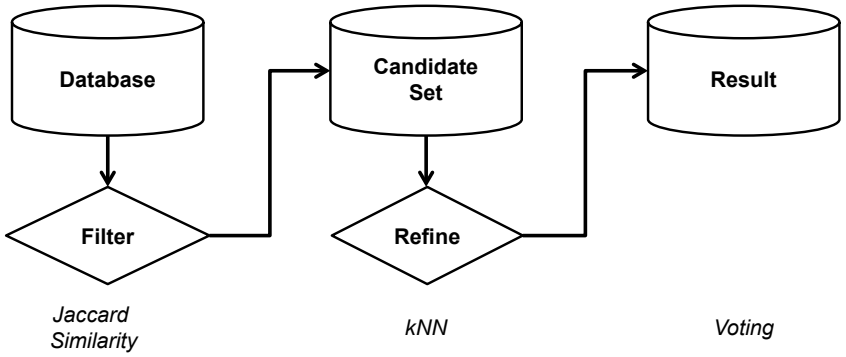

Fig. 2. High-level architecture of the proposed system

D. Trajectory Prediction

Due to the trajectory classification, the measured time series belong to a certain class containing one or more paths. The paths from this class can then be used to infer about the situation of the user and provide candidates for future movement. Such a dataset is represented in Figure 1 showing several locations within a building and bidirectional paths between them.

The correctness of the result can be statistically evaluated by probability calculations. If one path was more frequented in the past than others, the user will probably choose this path again. The more frequented paths are illustrated by a higher line width in Figure 1. Considering all possible locations, the user's most probable next destination can be determined based on the user's current path. Furthermore, using the conditional probability, we can identify a potential destination Des_C assuming that the user is currently on the path from A to B:

$$P(Des_C|Path_{A\to B}) = \frac{P(Path_{A\to B}|Des_C)\cdot P(Des_C)}{P(Path_{A\to B})}$$

In this expression, the right hand factors can easily be estimated from relative fractions in the dataset. This expression will then be evaluated using all destintations Des_C. Therefore, the denomitator is constant and can be left out and replaced by a normalization afterwards.

IV. Trajectory Similarity from Time Series

In order to find trajectories from the dataset, which are similar to a requested time series, we propose the following architecture, illustrated in Figure 2. The requested time series is preprocessed and taken as input. The result of our system is an amount of hits while each element is connected to a distance value indicating the similarity between the requested time series and a specific trajectory. The system's process is divided into three major steps:

- *Preprocessing:* The requested time series is preprocessed in order to remove needless samples and thus, to increase the performance in the following steps.
- *Filtering:* Based on the Jaccard Similarity, a large fraction of the dataset can be pruned.
- *Comparison:* The query trajectory is then compared to the candidate set using the k-NN approach. From these top k hits, voting approaches can be used to infer the most probable findings.

A. Preprocessing

In a fist step, redundant and needless information in the requested time series is removed from the total amount in order to improve the performance of subsequent processing steps. For that purpose, the requested time series is analyzed according to three features:

- *Multi-SSID:* For technical or organizational purpose, a single access point can manage various networks with different SSIDs and BSSIDs. Such multi-SSIDs can be summarized to one logical SSID in the recorded time series to minimize the amount of collected data. Different BSSIDs from one single physical access point can be identified due to the fact that multi-SSIDs often use similar or continuous BSSIDs. Furthermore, the received signal strengths from multi-SSIDs have similar qualities and thus, they can be grouped to one signal strength which reduces the amount of collected data.
- *Thresholding:* During a trajectory, the received Wi-Fi signal strengths strongly varies in terms of the received power. If all received signal strengths from one network in a complete time series are smaller than a predefined threshold, this network can be removed from the collected data. This reduces the amount of comparisons needed for classification. However, even a weak network can contain useful information for classification, which will be lost by this step. Hence, the threshold should be carefully chosen.
- *Mirroring of tracks:* Due to the fact that the direction of travel in time series is dispensable, a track contains the same information as its inverse. Hence, only one track for both directions must be saved.

Once redundant information is removed from the requested time series, we can simplify the remaining dataset by using the Douglas-Peucker algorithm. The aim of this step is to return trajectories, which are less complex than the original ones, but without loosing important information, e.g. maxima and minima. Hence, choosing the right value for the Douglas-Peucker parameter ϵ is essential to meet this requirement. If ϵ is too small, the complexity of requested time series is

not reduced to an optimal level and further processing steps waste resources. On the other hand, a high value for ϵ leads to information suppression and increases the error rate for classification.

Besides simplification, the requested time series can also be filtered from noise by using the FFT. Thus, trajectories are smoothed and high frequency noise is removed. This technique leads to information degradation of the requested time series and consequently, it has to be investigated, whether FFT-based denoising complicates classification.

B. Filtering

The purpose of this step is to reduce the amount of the dataset to be considered in order to reduce computational overhead and improve classification performance. Based on the assumption that the requested time series contains signals from the same access points as the correct trajectory from the data set, we can ignore tracks showing a small similarity with the requested time series in terms of existing access points. For filtering, the Jaccard-index with respect to the sets of fingerprints in each trajectory is being used. If q represents the requested time series and T_m represents one track out of the training dataset, we define the Jaccard-index J as

$$J(\mathrm{APs}_q, \mathrm{APs}_{T_m}) = \frac{|\,\mathrm{APs}_q \cap \mathrm{APs}_{T_m}\,|}{|\,\mathrm{APs}_q \cup \mathrm{APs}_{T_m}\,|}$$

with APs_x representing the amount of scanned access points on trajectory x. Hence, $J(\mathrm{APs}_q, \mathrm{APs}_{T_m})$ describes the similarity as the rate between the amount of common access points and the total amount of scanned access points in both tracks q and T_m. Each track T_m can then be classified according to its similarity to the requested time series q, based on the Jaccard-index and a predefined threshold δ. Only tracks with $J(\mathrm{APs}_q, \mathrm{APs}_{T_m}) > \delta$ are declared as similar to q and are put into the candidate set for further processing. Thus, choosing an optimal value for δ is essential for classification. If δ is suboptimal, e.g. too small or too high, the result of the classification process will degrade. Note that it is possible to choose δ in a data-driven and adaptive way by defining the relative fraction of the x% most similar tracks out of the dataset.

One problem with respect of performance and computing time is the arising overhead when calculating the intersection of existing access points between q and each T_m. As the information about existing access points in T_m is available before calculating the intersection for the Jaccard-index, we can reduce the complexity by computing a sparse matrix A. Then, the Jaccard index can be calculated using a sparse matrix-vector multiplication. Each column of A represents a trajectory T_m of the dataset and each line represents a specific access point AP_n. If an access point AP_n occurs in a track T_m, the value at $a_{m,n}$ is set to one. If not, then $a_{m,n}$ is set to zero.

$$A = \begin{pmatrix} a_{T_1,\mathrm{AP}_1} & \cdots & a_{T_m,\mathrm{AP}_1} \\ \vdots & \ddots & \vdots \\ a_{T_1,\mathrm{AP}_n} & \cdots & a_{T_m,\mathrm{AP}_n} \end{pmatrix}$$

$$\text{with } a_{T_m,\mathrm{AP}_n} = \begin{cases} 1 & \text{if } \mathrm{AP}_n \in T_m \\ 0 & \text{otherwise} \end{cases}$$

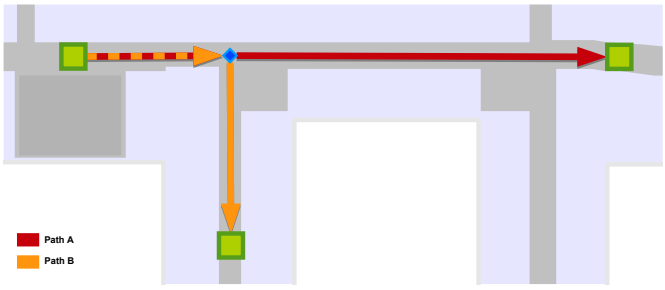

Fig. 3. Example showing two paths which share a segment.

For each requested time series q the size of the intersection of APs_q and APs_{T_m} can efficiently be computed by a sparse matrix-vector multiplication of A with the vector q_{AP_n} representing all exiting access points in q.

$$\begin{pmatrix} |\,\mathrm{APs}_q \cap \mathrm{APs}_{T_1}\,| \\ \vdots \\ |\,\mathrm{APs}_q \cap \mathrm{APs}_{T_m}\,| \end{pmatrix}' = \begin{pmatrix} q_{\mathrm{AP}_1} \\ \vdots \\ q_{\mathrm{AP}_n} \end{pmatrix}' \times A$$

To obtain the Jaccard-index for each entry, all intersections have to be divided by the total amount of scanned access points which is easy to compute.

C. Elementary Trajectory Comparison

After filtering the dataset based on the Jaccard-index and a predefined threshold δ, only a certain amount of trajectories remains as candidates for comparison with the requested time series. For each candidate, we compute the distance to the requested time series. Thus, elementary trajectory comparison is based on one of the distances introduced in Section III-A. The Hausdorff-distance considers measured signal strengths without their temporal relation. In comparison to Hausdorff, the Fréchet distance was implemented for time series and respects also the time domain of samples. DTW focusses on the problem that various time series differ in their speed of motion. Considering the distances from elementary trajectory comparison, we classify the requested time series by k-NN classification and return the k most similar tracks from the candidate set.

One problem with this approach is that different tracks can share large fractions of similar movement. Hence, when the requested time series is compared to the training dataset, it is customary to return a set of the top k trajectories. Additionally, these top k trajectories can be scored using the inverse of the distance of the trajectory candidate from the query trajectory.

D. Segmentation

For efficient comparison of trajectories of different length, we use a segmentation approach. Segmentation divides measured time series in different parts based on certain segmentation points. The aim of this step is to become able to find similar pieces in both, the requested time series and the dataset. Furthermore, a query with a short trajectory will then match to extensions of itself once the first segments have been completed. Figure 3 depicts a situation in which two paths start with the same segment and split into two different paths at the corner.

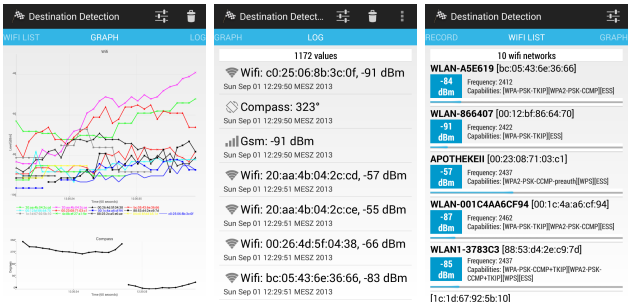

Fig. 4. Screenshots of data collection using the prototype

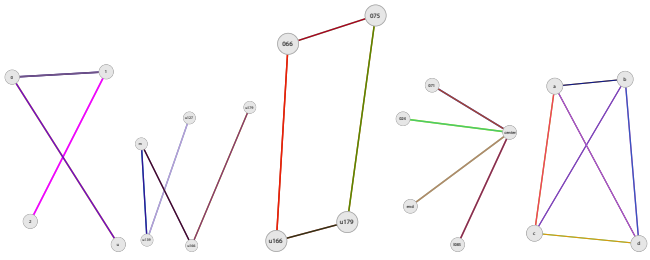

Fig. 5. Characteristics of the generated Datasets

There are different strategies to find such segmentation points in time series. We discuss the following strategies out of which segmentation at local maxima performs best on our datasets.

- *First occurrence of AP:* When a users enters the range of an access point for the first time, signals from this access point are first received. The beginning of signal measurements from a new access point can be used as a segmentation point.
- *Last occurrence of AP:* Similarly, we can define segmentation points, when an access point disappears.
- *Local maxima of AP inside trajectory:* Another way to define segmentation points is given by looking for local maxima of signal strength readings. It is reasonable to expect the user being at the nearest distance to the given access point.
- *High absolute variation:* Certain events, such as floor changes, passing doors or changing directions cause significant modifications of collected time series. Thus, segmentation can be performed, when received signal strengths variation increases extremely between two points in time.
- *High relative variation:* The absolute variation strategy can be extended to cover relative variation by dividing the absolute variation by the number of visible access points.
- *Thresholding*: This strategy is based on a predefined threshold. Segmentation points are placed whenever the received signal strength of an access point exceeds the predefined threshold for the first time.

V. Data Set and Implementation

In order to evaluate the choices involved in the proposed system and in order to get an understanding of the performance of the system, three steps have been taken. First of all, a prototype based on the Android mobile operating system has been developed. This prototype implements all major aspects of the system including data collection as well as database construction, labelling, and live predicition.

A. Prototype

The propotype consists of an Android application, which can be used to collect the needed dataset. It was installed on several private Android devices and collected Wi-Fi signal strength data by actively scanning for available access points once a second. This data can be recorded and labels for the starting point and the end point can be assigned. Figure 4 depicts three screens of this application, a graph containing historical Wi-Fi data, a list of sensor events captured as well as a list of access points.

B. Data Sets

For the evaluation of the proposed approach, we collected six datasets inside our office building. The first dataset is given by tracking the movement of a single person over some weeks inside our building. The dataset is subsumed in Figure 1 and contains a lot of different locations given by room names and varying numbers of movements between these rooms. This dataset has been used to evaluate the live prediction module inside the app.

In order to understand the situations, in which our direct approach is working well, we collected five additional tracks for the movement patterns depcited in Figure 5. Table I subsumes the contents of these datasets.

VI. Evaluation and Results

This section evaluates the three key ingredients to the proposed approach isolated from each other. First, we consider the impact of the choice of the elementary distance function measured by classification accuracy. Then, we discuss the flexibility of the approach by using the datasets of specific movement pattern, with respect to the choice of the parameter k. Finally, we discuss the impact of the segmentation approach on the classification accuracy. In summary, we showed, that the proposed architecture is flexible, and works as expected.

A. Impact of the Distance Measure

The comparision of the final classification performance of the proposed approach provides with the insight that the Fréchet distance best covers the characteristics of Wi-Fi trajectories in our datasets. While the DTW distance provides very good results for short fragments of the query trajectory, it is soon outperformed by the Fréchet metric, which is the only metric which reaches above 90% classification accuracy in the end. Figure 6 depicts these results.

Movement Pattern	Paths	Places	Trajectories	Average Duration	APs	Average Number of APs
Stairs	6	4	48	10,63 s	13	9,13 (≙ 70,23 %)
Hallway	8	5	64	27,98 s	43	15,25 (≙ 35,47 %)
Floor	8	4	40	23,22 s	20	12,75 (≙ 63,75 %)
Plus	8	5	64	23,21 s	38	14,78 (≙ 38,89 %)
Square	12	4	64	15,86 s	27	17,56 (≙ 65,04 %)

TABLE I. CHARACTERISTIC PROPERTIES OF THE DIFFERENT CASES.

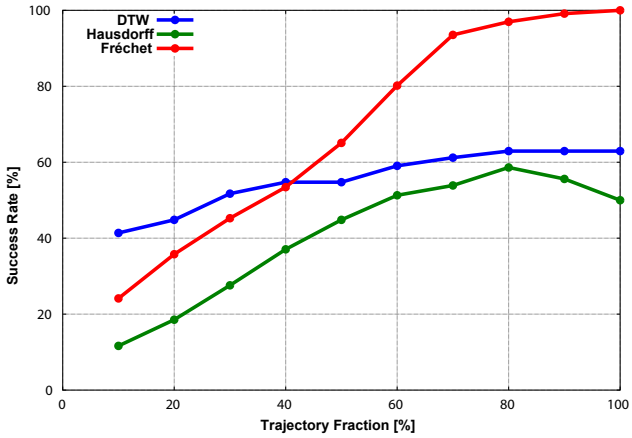

Fig. 6. Comparision of different distance functions

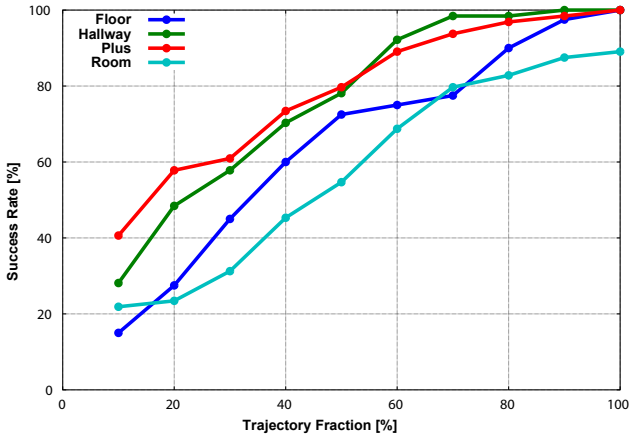

Fig. 7. Comparision of testcases

B. *Classification Results*

The choice of testcases for the evaluation and the performance of the propsed approach using the Fréchet metric in all cases is depicted in Figure 7. We can conclude from this setting, that movement patterns with high variation of signal conditions, such as walking a plus pattern at an indoor crossing or walking a hallway, have better results. This is to be expected, as the movement introduces dissimilarity. The approach directly exploits these signal fluctuations, which are problematic for a coordinate based position determination. In total, the system si working as expected in all cases.

In order to justify the additional complexity of selecting the top k-next neighbors in the dataset, we evaluated the result sets

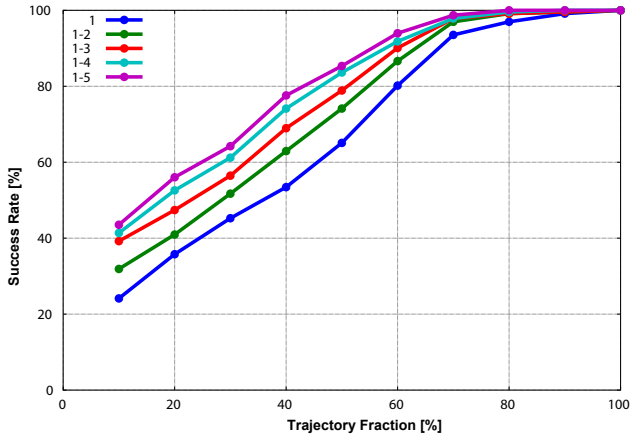

Fig. 8. Probability of finding the right trajectory in the top k nearest neighbors for varying k

for varying k. Figure 8 depicts this results.

Furthermore, we have to check that the simplification using Douglas-Peucker algorithm did not negatively affect classification performance. Fortunately, this is almost the case. Figure 9 depicts the performance of the system with and without Douglas-Peucker simplification.

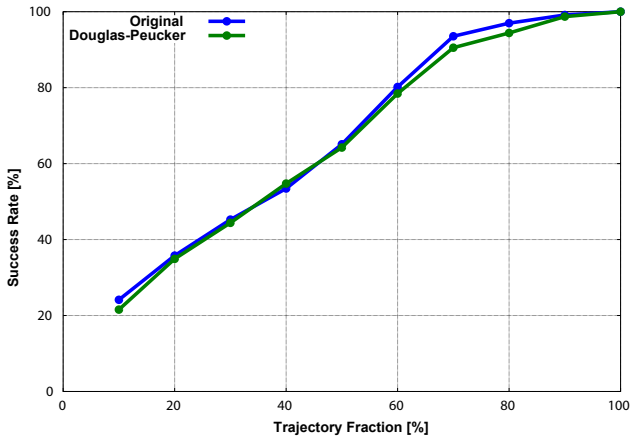

Fig. 9. Negative impact of Douglas-Peucker simplification

Note that the computational overhead as well as the communication cost can be greatly reduced by applying Douglas-Peucker algorithm and that the marginal negative impact is neglectible.

C. Segmentation

In order to evaluate the impact of the segmentation scheme, we evaluated the system for varying k and all six segmentation schemes of Section IV-D. Figure 10 depicts the success rate for the varying segmentation schemes.

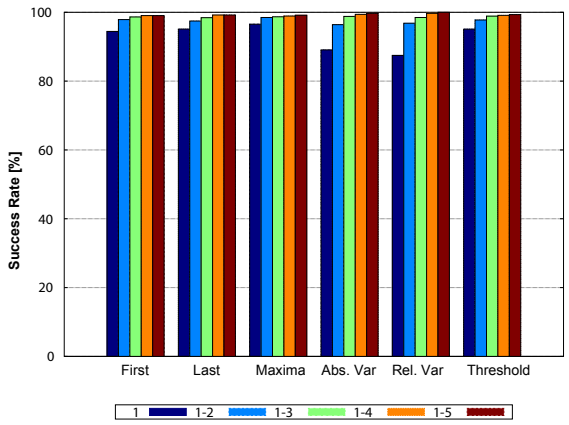

Fig. 10. Impact of segmentation schemes

The Y-axis measures the rate with which the correct trajectory has been found in the top k trajectories for $k = 1 \ldots 5$. It can be clearly seen that the maximum segmentation provides the best results for varying k. However, for larger k the difference is reduced and can be neglected for $k = 5$. In other words: In order to have the correct trajectory among the top five trajectories, the selection of segmentation scheme is irrelevant. However, the ordering of the top five trajectories is very sensitive to the choice of segmentation schemes.

VII. Conclusion

With this paper, we have provided a multilevel architecture for indoor trajectory processing using Wi-Fi signals as information source. We showed that it is possible to use Wi-Fi signals directly to infer the trajectory out of a given database of trajectories with high success rates. The results for the different test cases give a deep understanding of why signal anomalies, which are problematic for coordinate-based positioning, are useful for trajectory identification. This amounts to cases, where the signal quality quickly changes while traversing corners or staircases. One surprising result is that trajectory distance calculation based on dynamic time warping performs worse than classical monotonous distance processing given by Fréchet distance. Further, we conducted experiments with different ways of segmenting trajectories in an unsupervised manner only from sensor data. The results are promising and lead to smaller elementary trajectories and automatically allow for the detection of common subsequences of this type in different trajectories. Altogether, we have shown that it is possible to provide trajectory matching services using sensor data from mobile devices and that a top k candidate approach leads to very good results. The domains of possible application range from target guessing over proactive navigation to context-aware networking.

References

[1] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell, "Nextplace: a spatio-temporal prediction framework for pervasive systems," in *Pervasive Computing*. Springer, 2011, pp. 152–169.

[2] J. A. Álvarez, J. A. Ortega, L. G. Abril, F. Velasco, and F. J. Cuberos, "¿ where do we go? ontheway: A prediction system for spatial locations." in *ICUC*, 2006.

[3] J. Froehlich and J. Krumm, "Route prediction from trip observations," *SAE SP*, vol. 2193, p. 53, 2008.

[4] T. M. T. Do and D. Gatica-Perez, "Where and what: Using smartphones to predict next locations and applications in daily life." in *Pervasive and Mobile Computing*, 2013.

[5] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "Mining user mobility features for next place prediction in location-based services," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 1038–1043.

[6] J. Krumm and A. B. Brush, "Learning time-based presence probabilities," in *Pervasive Computing*. Springer, 2011, pp. 79–96.

[7] J. Wang and B. Prabhala, "Periodicity based next place prediction," in *Nokia Mobile Data Challenge 2012 Workshop. p. Dedicated task*, vol. 2, no. 2, 2012.

[8] L.-H. Tran, M. Catasta, L. K. McDowell, and K. Aberer, "Next place prediction using mobile data," 2012.

[9] Z. Lu, Y. Zhu, V. W. Zheng, and Q. Yang, "Next place prediction by learning with multiple models."

[10] J. Petzold, F. Bagci, W. Trumler, and T. Ungerer, "Comparison of different methods for next location prediction," in *Euro-Par 2006 Parallel Processing*. Springer, 2006, pp. 909–918.

[11] D. Katsaros and Y. Manolopoulos, "Prediction in wireless networks by markov chains," *Wireless Communications, IEEE*, vol. 16, no. 2, pp. 56–64, 2009.

[12] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive wi-fi mobility data," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2. IEEE, 2004, pp. 1414–1424.

[13] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility markov chains," in *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*. ACM, 2012, p. 3.

[14] H. Alt and M. Godau, "Computing the fréchet distance between two polygonal curves," *International Journal of Computational Geometry and Applications*, vol. 5, no. 1, pp. 75–91, 1995.

[15] T. Eiter and H. Mannila, "Computing discrete fréchet distance," 1994.

[16] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[17] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[18] Y. Zheng and X. Zhou, *Computing with Spatial Trajectories*. Springer, 2011, vol. 308.