Identifying movements in noisy crowd analytics data

Cristian Chilipirea, Ciprian Dobre
University Politehnica of Bucharest
Romania
{cristian.chilipirea, ciprian.dobre}@cs.pub.ro

Mitra Baratchi
Leiden University
The Netherlands
m.baratchi@liacs.leidenuniv.nl

Maarten van Steen
University of Twente
The Netherlands
m.r.vansteen@utwente.nl

***Abstract*—Privacy-preserved tracking of WiFi-enabled devices such as smartphones offers a highly scalable solution for large-scale crowd movement studies. However, extracting knowledge out of pedestrian-tracking data acquired this way is not simple. This is, generally, due to the inherent inaccuracy of the measurement technique.**

Segmenting an individual's trajectory data into periods of *stops* and *moves* is a fundamental step in analyzing crowds' movement. Such distinctions allow us to answer advanced questions regarding visited locations or even social behavior. Algorithms previously designed for distinguishing movements from stay periods, assume datasets are gathered using GPS, which offers precise positioning. WiFi tracking, however, does not offer such precision. The location of devices can at best be reduced to a large area around the WiFi scanner.

In this paper, we study a set of established algorithms for detecting periods of stops and moves from GPS-based datasets and their applicability to WiFi-based data. Consequently, we propose possible improvements to such algorithms considering the inherent characteristics of WiFi tracking data.

Keywords: tracking data, trajectory data mining, WiFi tracking, mobility modeling

I. Introduction

When taking a birds-eye view at the motion of a person in a crowd, it can be difficult to give it meaning. This is especially true when traversing tracking data, composed of a set of timestamped positions. These positions can be abstracted to periods of stops and movements [1], [2], building blocks with which we can answer many questions: "Where do people go?" "How long do they stay there?", "Can we infer social relationships?", "What can we do to stimulate certain behavior?", and so on.

In this paper, we concentrate on monitoring crowds of people through Radio Frequency-based (RF-based) detection of on-person devices, such as those using WiFi or Bluetooth, and address the problem of separating stops from movements in an individual's trace. Since RF-based scanning introduces considerable noise, identifying stops and movements in such data sets is a nontrivial problem.

The problem has previously been solved for tracking datasets containing many accurate positions of an individual (GPS traces). A GPS device records positions along the path of an individual with relatively high precision. Stops appear as positions randomly placed within a restricted area. Clustering methods are used to group such positions and obtain stop periods. We have identified three methods: CbSmot [2], Dbsmot [3] and Stay Point Detection [4] or [5] that are previously defined for this purpose. These methods make use of different properties of a trace: speed, direction and respectively distance.

For city-scale crowd monitoring, tracking WiFi-enabled devices [6] is the preferred method. The method assumes that most people carry a WiFi-enabled device (e.g., a smartphone) and by recording elements of the network frames transmitted by these devices, according to the 802.11 standard, we can track large crowds at a small cost. Except from having WiFi enabled, the approach does not require active participation from people in the crowd (which is necessary for GPS traces), making it easy to deploy.

Unfortunately, for outdoor environments, WiFi tracking datasets are often sparse and affected by various sources of noise. This noise increases when trying to monitor large crowds of people (caused by network congestion and the fact that people obstruct RF signals). In the case of a device that is sitting still between multiple sensors the noise in the data can easily make it seem to have an erratic, moving behavior between those sensors. In ideal circumstances, the device would be detected by all sensors with every frame it sends, but we have found this is rarely the case. When a device is moving, the noise has an even stronger effect: a device can be first detected by the sensor in front of it and then by the one behind; Making it impossible to identify the direction of movement from just two detections.

It is possible to place sensors far enough, so that erratic behavior is not present. However, placing sensors such that an area is fully covered *and* such that they do not have overlapping detection areas is impossible. Even in ideal circumstances geometry doesn't permit non-overlapping discs of the same size to completely cover an area. A sensor's detection range is not only irregular and difficult to accurately determine, but also changes due to uncontrollable conditions, such as weather.

To our knowledge, we are the first to test the accuracy of the methods for separating static from mobile periods on datasets gathered using RF-signals (WiFi) detections. Some previous work has managed to identify movements indoor based on information that we found to be too erratic, received signal strength indicator (or RSSI) [7]. Furthermore, we show that, for these type of datasets, even a method that would label

the detections perfectly cannot reach achieve accuracy. This is because of the low detection rate of the tracking method.

After we identify the method that provides the best results (Stay Point Detection) we search for ways of improving its accuracy. We experiment with different distance metrics obtained from the tracking dataset itself.

II. WiFi tracking datasets and their noise

WiFi tracking data is gathered by using several sensors. These sensors are configured to record the time at which they receive an 802.11 frame (usually a probe request) along with the anonymized id (hash of the MAC inside the frame) of the device sending it, used to differentiate between different devices. The location is added in the form of a sensor id. The location of sensors is known and by using the sensor id we can determine an approximate position of the transmitting device.

Sensors capture frames from signals that reach them and can be correctly interpreted. The WiFi protocol is designed to transmit data at a range of about 100m. However, weather, geography, interference with the environment, as well as many other factors influence the distance and the shape of the area in which a frame can be received. In contrast, GPS datasets have an accuracy in the order of few meters.

A large variation in software, hardware and manufacturing techniques adds to the noise already generated by the environmental effects on the WiFi signal. These differences affect the frequency with which probe requests are sent. More so, there can be multiple probe request frequencies for the same device. This change has been analyzed previously [8] and it has been shown that even the state of the screen (whether it is on or off) has an influence. We should also expect frequency changes based on the battery level or even installed applications. In contrast, for GPS the recording frequency can be very high and constant over time.

Detection frequencies can be improved by placing sensors inside buildings. In the work of [1] the authors claim to have 90% of detections with less than one second between them. Our dataset, gathered outdoors, during a large gathering, has only 20% of detections with less than one second between them.

To acquire large scale data, we performed a data collection experiment during a festival that gathers more than 150,000 people. When crowds of this size gather in a small area, WiFi quality becomes an issue, as control frames can easily cover a large part of the bandwidth. Because of this, many probe requests are lost due to collisions.

To illustrate how noisy our WiFi tracking dataset is, we extracted a few detections from one device. This device was selected because it has a very large number of detections at only three sensors and is present throughout the entire scanning period (meaning it is most likely immobile). The device has an almost equal number of detections at two of the sensors (about 30,000 each) and very few at the third. This leads us to believe the device is located between two of the sensors, let's call them A and B.

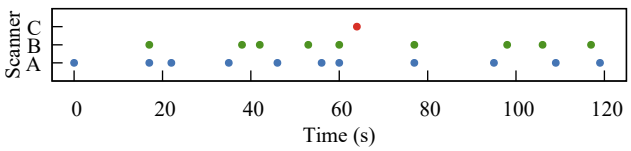

Fig. 1: Noisy Detections

In Fig.1 we can see that the device is rarely detected by both sensors at the same time, seeming to move back and forth between the two. A similar behavior has been observed in [9] where it was referred to as "ingpong effect. Less than 4% of detections are recorded at both sensors simultaneously and less than 0.5% have the same sequence number, meaning they are detections of the same network frame.

We found that for this device, the RSSI values (ranging from -20 to -80) have a high variation. The RSSI measures the strength with which a signal is received, in theory it correlates with the distance between the sender and receiver. We calculated the standard deviation and obtained a value of eight. Even worse, for the rest of devices that we identified as immobile, the standard deviation is equal to nine. This shows how inaccurate RSSI can be. Because of this and the manufacturing differences, the RSSI value cannot be used to increase the positional accuracy of detections.

The RSSI value of each frame is more accurate if the sensors are placed inside a room [10]. The authors measure relationships based on how much time people spend together inside a dining hall. The authors use RSSI values for multilateration to determine which people dine together. Multilateration requires simultaneous detections. We note that the authors claim 95% of their detections have a frequency of under two minutes, while we only have 83% of our detections with less than two minutes between them.

III. Related Work

There has been a large amount of research analyzing WiFi traces to extract information on crowds. Initially, this information consisted mostly on determining the size of the crowd [11] and verifying how it can be used to estimate the actual number of people. From an analysis perspective, there is a hard limit on how much information we can gain by simply counting.

More advanced information can be extracted by using domain knowledge. This is shown in [12] where they use domain knowledge on schedules and behavior or hospital personnel and paths of people inside a hospital to determine facility planning. However, domain knowledge is often limited and offers only application-tailored solutions that rarely work in the general case.

Analysis can be performed at low resolution, avoiding the need to deal with noise. This is the case of [13] where people are grouped based on several factors: SSIDs sent with the probe requests (we avoid the use of SSIDs as they divulge private information [14]), previous knowledge on SSIDs and, binary

matrices, built for every individual, with rows representing locations and columns representing time slots. These time slots are set to one hour.

Analysis of movement data offers a large potential. The authors of [15] managed to identify not only relationships, but the strength of relationships both for humans and animals. Even more, [16] showed they can identify groups by simply matching the times at which people enter and exit a room. This is in-line with the same reasoning we have for separating moves from stops.

We based our experiments on datasets that have a lot in common with the one from [17], in which the data is gathered during a multi-stage festival in Denmark. Their dataset is gathered with a different RF protocol, namely Bluetooth. Bluetooth functions at a smaller range, resulting in an increase in accuracy and noise reduction. However, because Bluetooth is turned off by default in most consumer products, their dataset has fewer detections. For a festival of a similar size with the one we analyze they detect ten times fewer devices.

IV. Detecting Movements

The raw tracking data consists of a set of timestamped locations. Due to the sparsity of data, and considerably long-time intervals between detections, even when represented on the map, in time, it is difficult to make sense of the data in this raw form.

There has been a large amount of research on making sense of GPS data, and a vital processing step is to extract static/-movement periods from the dataset. Having this separation permits one to detect flows, places of interest and it sets the groundwork to be able to answer more complex questions.

A major difference between GPS and WiFi-based movement data is the precision level. While GPS data has an accuracy of 5 meters or less, WiFi detection range can reach 200 meters.

We identified three algorithms [2], [3], [4] used to separate GPS detections in static and movement periods. These algorithms use clustering techniques to group detections that are close to each other. The clusters found represent periods when the device is static or inside a building. Points that do not fall into any cluster represent movement. This approach performs well on GPS data because when a device is static most of the detections fall within a small neighborhood. Using WiFi tracking this neighborhood would be considerably wider. We applied these algorithms on WiFi datasets to inspect how well they perform. The main difference in implementation would be to replace the WiFi sensor ids in the dataset with their position.

More specifically, these algorithms are:

- Cbsmot [2] has its base in the dbscan algorithm. It forms clusters only of consecutive detection and considers both distance and time when building the clusters. The interesting use of time and distance translates in clusters forming when the speed is low.
- Dbsmot [3] is a variation and extension of dbscan. It forms clusters by taking consecutive detections and considering the change in direction. The assumption is that if a device is static it appears to change direction frequently, as detections are positioned randomly around it. In contrast, during movement the same direction is kept.
- Stay Point Detection [4] or [5] is one of the simplest algorithms, it makes an intuitive assumption, if one of the following detections is further than a threshold from a pivot location, the person must have moved. With each movement the pivot gets updated.

The algorithms we have chosen, make use of different attributes of movement which can be extracted from trajectory data (distance/speed/direction). By comparing their performance, we can see which one better fits the characteristics of our WiFi generated dataset.

V. Method Comparison

A. Dataset

We have placed 40 WiFi sensors in the city of Assen, The Netherlands during the TT Festival[1] and collected WiFi tracking data for each festival day in the years 2015, 2016 and 2017. Details on the datasets can be found in [6]. The results presented in this paper are based on the data from 2016.

The sensors were configured to capture any probe requests frame they received. Once a frame is received a detection is created with a time stamp t, the device id d (in the form of a salted hash of the MAC address inside the probe request frame), the id of the sensor s that received it and the sequence number n extracted from the frame.

B. Comparison

To compare the algorithms, we use a part of the dataset for which we have ground truth. During the Assen 2016 festival we walked around the area carrying nine WiFi enabled devices (phones and tablets). We recorded GPS positions from the devices as well as made notes on our movements. Out of these positions we manually construct a list of times when we are moving and times when we are static. The resulted list represents the ground truth.

The algorithms we use have multiple parameters: Cbsmot has an *maximumDistance* and *minimumTime* which can be set to control a maximum speed; Dbsmot has a value that represents the *minimalDirectionChange* and the minimal size of a cluster; Stay Points requires the setting of only a *minimalDistance*, as large as to be considered a movement. After static and movement periods are separated we remove all static periods which are shorter than a *mergeTime* value. These short periods are not significant. To explain, consider a person going by bus to a shop. We are interested on the big activities such as shopping, but ignore possible unintended ones, like the bus stops. The small stops would clutter our results and make it more difficult to extract knowledge.

We use the three methods to extract static/moving periods for these nine devices. We then compare the extracted periods with the ground truth. Having only two classes, this could be considered as a binary classification problem. We mark movement as positive and stopping periods as negative. We

[1] https://www.ttfestival.nl/en/ (27-June-2017)

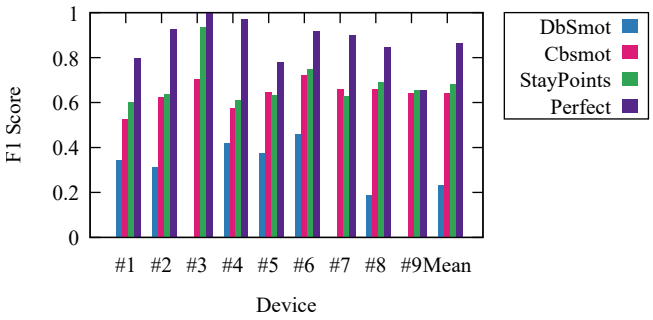

Fig. 2: Comparing algorithms

count the seconds when the periods match and mark them as true positives/negatives (TP/TN). The same is done for false positive/negatives (FP/FN). Using these values, we calculate the F1 Score for each of our nine devices using the formula in equation 1. The F1 score represents an accuracy measure that considers both precision and recall. We used multiple values for each of the knobs presented above and selected the settings for which the mean F1 score is the highest. The results can be seen in Fig.2.

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{1}$$

As seen, the least accurate method is Dbsmot. The small number of possible positions can explain this. Meaning that direction changes are more frequent than in the case of GPS, making static/movement separation difficult based on the angle of movement.

Cbsmot and Stay Points offer similar results, however Stay Points is much simpler and runs significantly faster.

We add "perfect" to our results which is calculated by using ground truth to mark detections as static or mobile. Because detections have significant time gaps even a perfect labeling cannot reach a F1 score of one.

VI. Improvements on the distance function

The most accurate and simplest method to separate static periods from moving ones is Stay Points. We aim to improve this method by changing the distance function. In the original algorithm, the distance function is the geographical distance.

The geographical distance is not ideal if we consider people move inside a city. Their movement is limited to the network of streets. Furthermore, the buildings block or even increase the range of the WiFi signals, creating tunnel effects. This means that two locations that are close geographically could be separated by large buildings.

Considering that the set of sensors is limited there is only a limited set of distances that need to be considered. By applying a distance function on all sensor pairs, we obtain the set of required distances. They can be modeled as a graph that has sensors as nodes and where each edge has a weight corresponding to the distance between the two sensors at its end.

A. Definitions

We use the following notations:

S - the set of sensors $\{s_1, ...s_N\}$ where one sensor is s_p and a subset of sensors is $s_\mathbf{P}$

D - the set of devices $\{d_1, ...d_M\}$

t - time, measured in seconds

Λ - the set of detections $\{\lambda_1, ...\lambda_R\}$ where a detection is a tuple $\langle s, d, t, n\rangle$ (where n is the sequence number of the request)

$\Lambda[s]$ - is the set of detections at sensor s. Analogous notations $\Lambda[d]$, $\Lambda[t]$, $\Lambda[n]$. Similarly $\Lambda[d][s]$ represents the set of detections of device d at sensor s.

$\overline{\Lambda} = (\lambda_1, ...\lambda_R)$ - represents a sequence of detections so that they are ordered by device, time, sequence number and finally sensor.

$\overline{\Lambda[s]} = (\lambda_0 ...\lambda_P)$ - represents a sequence of consecutive detections at sensor s. It is a subset of $\overline{\Lambda}$ in which $\forall i; \lambda_i[s] = s$ Analogous notation for $\overline{\Lambda[d]}$.

B. Sensor Neighborhood Graphs

We build the sensor distance graph (SDG). Let $SDG = (S, E)$ where each node is a sensor and edges have weights associated with them based on a distance function. Furthermore, let ε be a threshold that removes all edges with a weight higher (or lower - depending on the distance function) from the SDG. We call this new graph the Sensor Neighborhood Graph (SNG). The basic idea is that we identify movement if a device is detected by two sensors that are not connected by an edge in the SNG.

We found a set of distance functions which can be used to build the SDG and SNG:

a) Geographical Distance (DIS): The weight of every edge is the geographical distance between the sensors at its ends. Smaller distances mean the sensors are closer, as such the SNG contains the edges with the weight smaller than ε.

$$E = \{(s_i, s_j, w_{ij}) | \forall s_i, s_j \in S; w_{ij} = \text{GPSdistance}(s_i, s_j)\} \tag{2}$$

b) Consecutive detections (CON): It is reasonable to assume that a person being still or moving between two sensors would generate consecutive detections, one at the first sensor and one at the second. We count the number of such occurrences for all pairs of sensors. Sensors close to each other have many such consecutive detections. The SNG is composed of edges where the weight is larger than ε. This remains unchanged for the next two methods.

$$\begin{aligned} C(s_i, s_j) = \{k | k \in [1, R]; \exists \lambda_k, \lambda_{k+1} \in \overline{\Lambda}; \\ \lambda_k[s] = s_i; \lambda_{k+1}[s] = s_j; \lambda_k[d] = \lambda_{k+1}[d]\} \\ E = \{(s_i, s_j, w_{ij}) | \forall s_i, s_j \in S; w_{ij} = |C(s_i, s_j)|\} \end{aligned} \tag{3}$$

c) Simultaneous detections (SIM): Consecutive detections in our dataset can have different timestamps. Here, we only count detections with the same time stamp. This means a device must be in range of both sensors.

$$\begin{aligned} C1(s_i, s_j) = \{k | k \in C(s_i, s_j); \lambda_k[t] = \lambda_{k+1}[t]\} \\ E1 = \{(s_i, s_j, w_{ij}) | \forall s_i, s_j \in S; w_{ij} = |C1(s_i, s_j)|\} \end{aligned} \tag{4}$$

d) Simultaneous detections validated with frame sequence number (SEQ): Our dataset has a resolution of one second. Because WiFi transmission frequency is higher, two different frames from a device can be simultaneously detected at two sensors. By recording sequence numbers, we can verify if the same frame is detected.

$$C2(s_i,s_j) = \{k | k \in C1(s_i,s_j); \lambda_k[n] = \lambda_{k+1}[n]\}$$
$$E2 = \{(s_i,s_j,w_{ij}) | \forall s_i,s_j \in S; w_{ij} = |C2(s_i,s_j)|\} \quad (5)$$

From the definition $C2 \subseteq C1 \subseteq C$. This means that the weights for $E2$ are smaller or equal to the weights of $E1$ and both are smaller or equal to the weights of E. Because the SNG contains only edges with high weights it would have fewer edges in $E1$ than in E and likewise for $E2$. A SNG with a small number of edges can be used to identify more refined movements. However, having too few edges brings the risk of identifying erratic behavior as movement.

Given the SNG, we modify the Stay Point algorithm to detect movements when we have detections at two sensors that are not connected by an edge.

VII. Improvement Experiments

We compared the graphs described in the previous section to determine which is best suited for determining movement. The graphs described are all complete, weighted graphs (have an edge between any pair of nodes). To detect movements, we need to find a threshold value, ε, for each of the graphs. We consider detecting a movement if the weight of an edge is larger (in the case of distance graph) or smaller (for the other graphs) than this ε.

We compare the graphs by keeping only a subset of edges. For graphs with zero edges, any noise in the dataset or any detections at two sensors are identified as movement. In contrast, when the graph is fully connected, we can't identify any movement and it appears like the devices are static. The number of edges depends on the application. For instance, when the purpose is to identify out-of-town trips we would want a graph that has edges between all sensors in the same city and no edges between sensors in different cities.

We sort the edges by weight (increasing for the distance graph and decreasing for the rest), start with empty graphs and add one edge at a time. We add the same number of edges to the four graphs and count how many edges are in common. The results can be observed in Fig.3. When the graphs are full they all have the same edges. Therefore, the percentages all converge to 100%. We added a random ordering of edges to have a benchmark with which to compare (Rand-Rand). For these graphs the percentage grows linearly with the number of edges.

The most similar graphs are the consecutive and the concomitant ones (CON-SIM). Consecutive detections contain all simultaneous ones and the simultaneous detections represent a third of them. The graph generated using sequence numbers is the closest to the distance graph (DIS-SEQ). This is a strong argument for the sequence number graph as it is intuitive

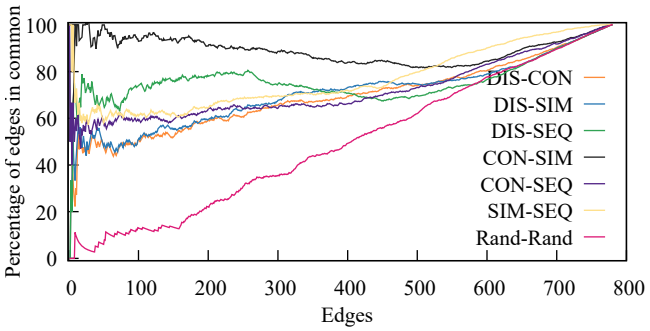

Fig. 3: Comparing graphs by counting edges in common

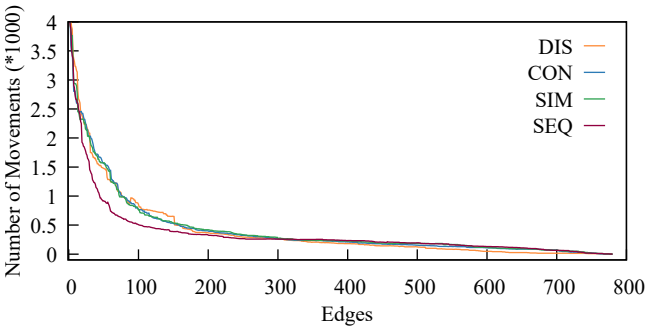

Fig. 4: Comparing graphs based on the number of movements

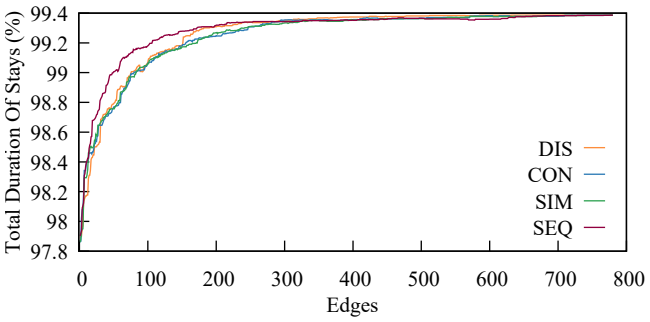

Fig. 5: Comparing graphs based on the total duration of stays

that geographical distance should play a primary role when considering movements.

We selected 100 devices from the entire dataset and ran Stay Point using the distance from the graphs. We added one edge at a time to see the effect. We then counted how many movements appear (Fig.4), as well as, how long the total duration of stays is (Fig.5). In both figures the graph generated using the sequence number stands out from the other three. We also note that the difference between the graphs is small.

To understand the accuracy of our algorithm we manually extracted two groups of 100 devices each from our dataset. One group is made only of mobile devices (M) and the other of static devices (S). We ran Stay Points on both groups using all four graphs and adding one edge at a time. We then counted how many devices our algorithm identifies as mobile (a device is considered mobile if it has at least one moving period). For

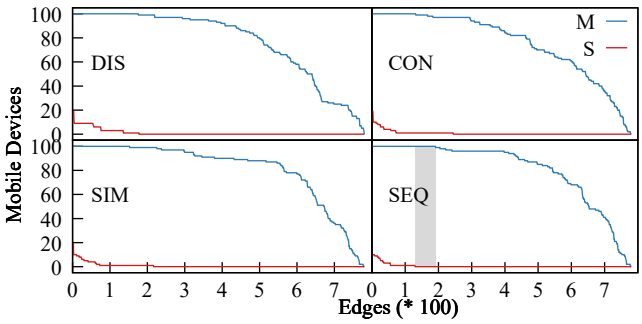

Fig. 6: Comparing graphs using Static and Mobile devices

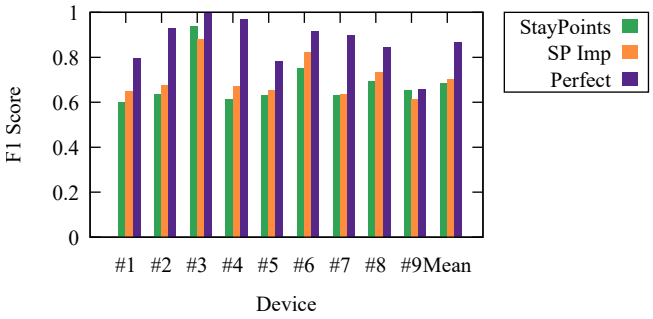

Fig. 7: Comparing algorithms

a perfect accuracy, the algorithm should identify all mobile devices as mobile and no static devices as mobile. The results are shown in Fig.6. The gray area represents the only part when perfect accuracy is reached. This is possible only for the graph created using sequence numbers.

We calculate the F1 score on our nine devices using the distances from the graph created with simultaneous detections with the same sequence number (SP Imp). We can see how the results compare to the original Stay Point in Fig. 7. Similarly with the previous results from Section V, we choose the setting that offers the best mean F1 Score. Instead of setting the *minimalDistance*, of Stay Point, we set the *numberOfEdges*.

VIII. Conclusion

Understanding movements in crowds is important. It opens the way to a multitude of applications and enables us to improve many things such as facility or urban planning.

The raw data needs to be processed. The first step is to identify periods of movements and separate them from the ones where a device is static. This permits us to answer more advanced questions.

We showed how algorithms that separate GPS data into static and moving periods work well with WiFi datasets. This is important, because WiFi datasets allow a more scalable solution for collecting datasets from crowd-movement.

Furthermore, we showed how we can use a distance function that uses solely the tracking data to further improve the accuracy of the best static/moving period separation method.

Acknowledgment

We thank Roel Schiphorst from BlueMark Innovations for providing us with the Assen data sets.

References

[1] F. Wang, X. Zhu, and J. Miao, "Semantic trajectories-based social relationships discovery using wifi monitors," *Personal and Ubiquitous Computing*, vol. 21, no. 1, pp. 85–96, 2017.

[2] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares, "A clustering-based approach for discovering interesting places in trajectories," in *Proceedings of the ACM symposium on Applied computing*, 2008, pp. 863–868.

[3] J. A. M. R. Rocha, V. C. Times, G. Oliveira, L. O. Alvares, and V. Bogorny, "DB-SMoT: A direction-based spatio-temporal clustering method," *2010 IEEE International Conference on Intelligent Systems, IS 2010 - Proceedings*, pp. 114–119, 2010.

[4] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello, "Extracting places from traces of locations," in *Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots*. ACM, 2004, pp. 110–118.

[5] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W. Ma, "Mining user similarity based on location history," *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, no. c, p. 34, 2008.

[6] A.-C. Petre, C. Chilipirea, M. Baratchi, C. Dobre, and M. van Steen, *Smart Sensor Networks*. Elsevier, 2017, ch. WiFi tracking of pedestrian behavior, pp. 309–336.

[7] T. King and M. B. Kjærgaard, "Composcan: adaptive scanning for efficient concurrent communications and positioning with 802.11," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 67–80.

[8] S. Jamil, S. Khan, A. Basalamah, and A. Lbath, "Classifying smartphone screen on/off state based on wifi probe patterns," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, 2016, pp. 301–304.

[9] M. Mun, D. Estrin, J. Burke, and M. Hansen, "Parsimonious mobility classification using gsm and wifi traces," in *Proceedings of the Fifth Workshop on Embedded Networked Sensors (HotEmNets)*, 2008.

[10] H. Hong, C. Luo, and M. C. Chan, "Socialprobe: Understanding social interaction through passive wifi monitoring," in *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 2016, pp. 94–103.

[11] L. Schauer, M. Werner, and P. Marcus, "Estimating crowd densities and pedestrian flows using wi-fi and bluetooth," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2014, pp. 171–177.

[12] A. J. Ruiz-Ruiz, H. Blunck, T. S. Prentow, A. Stisen, and M. B. Kjaergaard, "Analysis methods for extracting knowledge from large-scale wifi monitoring to inform building facility planning," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2014, pp. 130–138.

[13] N. Cheng, P. Mohapatra, M. Cunche, M. A. Kaafar, R. Boreli, and S. Krishnamurthy, "Inferring user relationship from hidden information in wlans," in *IEEE Military Communications Conference, MILCOM*, 2012, pp. 1–6.

[14] A. Di Luzio, A. Mei, and J. Stefa, "Mind your probes: De-anonymization of large crowds through smartphone wifi probe requests," in *The 35th Annual IEEE International Conference on Computer Communications INFOCOM*, 2016, pp. 1–9.

[15] Z. Li, B. Ding, F. Wu, T. K. H. Lei, R. Kays, and M. C. Crofoot, "Attraction and avoidance detection from movements," *Proceedings of the VLDB Endowment*, vol. 7, no. 3, pp. 157–168, 2013.

[16] N. Abedi, A. Bhaskar, and E. Chung, "Tracking spatio-temporal movement of human in terms of space utilization using media-access-control address data," *Applied Geography*, vol. 51, pp. 72–81, 2014.

[17] J. E. Larsen, P. Sapiezynski, A. Stopczynski, M. Mørup, and R. Theodorsen, "Crowds, bluetooth, and rock'n'roll: understanding music festival participant behavior," in *Proceedings of the 1st ACM international workshop on Personal data meets distributed multimedia*. ACM, 2013, pp. 11–18.