Presumably simple: monitoring crowds using WiFi

Cristian Chilipirea, Andreea-Cristina Petre, Ciprian Dobre
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
Bucharest, Romania
Email: {cristian.chilipirea, ciprian.dobre}@cs.pub.ro, andreea.petre@cti.pub.ro

Maarten van Steen
CTIT
University of Twente
Enschede, The Netherlands
Email: m.r.vansteen@utwente.nl

***Abstract*—Monitoring crowds is receiving much attention. An increasingly popular technique is to scan mobile devices, notably smartphones. We take a look at scanning such devices based on transmitted WiFi messages. Although research on capturing crowd patterns using WiFi detections has been done, there are not many published results when it comes to tracking movements. This is not surprising when realizing that the data provided by WiFi scanners is susceptible to many seemingly erroneous and missed detections, caused by the use of randomized network addresses, overlap between scanners, high variance in WiFi detection ranges, among other sources.**

In this paper, we investigate various techniques for cleaning up sets of raw detections to sets that can subsequently be used for crowd analytics. To this end, we introduce two different quality metrics to measure the effects of applying various data filters. We test our approach using a data set collected from 27 WiFi scanners spread across the downtown area of a Dutch city where at that time a 3-day multi-stage festival took place attended by some 130,000 people.

Keywords: crowd detection, WiFi scanning, crowd analytics, smoothing paths, data cleaning

I. Introduction

Being more an art than a science, managing crowds has proven to be important, yet difficult. Difficulties are partly caused by the lack of sufficient, properly validated models of pedestrian movements. As a consequence, automatically deriving accurate predictions based on input from various data sources is generally lacking. This leaves the experts to rely mostly only on their experience when it comes to preparing for an event and acting on monitored behavior. Providing automated decision support to crowd managers starts with data, yet relatively few data sets are (publicly) available on pedestrian movements in crowds. Although telecom providers do have access to movements at the level of cells, such data is not made generally available for various reasons, but one may also question whether the granularity is sufficient for validating models of pedestrian movements. Likewise, using data sets from video observations imposes serious privacy concerns, but also deriving data on crowd behavior from video sources that can be used for model validation is difficult, if not oftentimes virtually impossible.

In this paper we discuss a different approach, namely deriving data from detecting WiFi-enabled mobile devices such as smartphones. The idea is extremely simple: at an event, one places up to a few tens of WiFi scanners akin to normal access points, capable of sniffing network addresses. To provide some level of privacy, a network address is hashed before being further processed, thus ensuring that the owner of a smartphone cannot be identified if only the hashed value is available. The result is a data set consisting, conceptually, of timestamped (scanner, device) pairs. As such, this should allow us to discover movements as they take place in a crowd assuming we know the location of the WiFi scanners.

We recently conducted an experiment with such a setup during a three-day festival spread across the downtown area of Assen, a city in the Northern part of The Netherlands. The festival was visited by an estimated 130,000 people. Some 25 WiFi scanners were strategically placed to cover the locations of scheduled events, specific roads and junctions, and locations such as the train station and a camping site. Although we aimed to obtain information on crowd movements, we actually anticipated that even this seemingly simple approach would lead to a highly noisy data set. We were not disappointed.

Difficulties were caused by the use of short-living network addresses, nonmobile devices that exhibited intermittent detection intervals, seemingly erratic behavior when a device was in range of multiple scanners in a short timespan, the relatively low frequency at which a device is detectable, to name a few sources of errors. In this paper, we focus entirely on systematically cleaning the data set, making the statement that this is by far an easy exercise and essential before any analytics can take place. The main problem is to extract data on only mobile devices and in such a way that seemingly erratic and senseless detections are transformed or eliminated to come to useful information. To the best of our knowledge, we are one of the first to report on extracting crowd movements from actual real-world WiFi detections at this scale.

Our main contribution is formed by three distinct methods that improve the data set by removing low-quality detections, averaging detections over a time period and, eliminating repetitive behavior in the form of unlikely cycles in movement, respectively. Of course, we need to be able to evaluate the effectiveness of these cleaning methods. To this end, we define two metrics: (1) entropy, which measures how much noise there is, and (2) dissimilarity with respect to the original data set, which measures how much we modified the data set.

We organize our paper as follows: The next section presents the data set and the context in which it was collected; Section III presents related work regarding to this type of data cleaning; It is followed by Section IV which goes into the

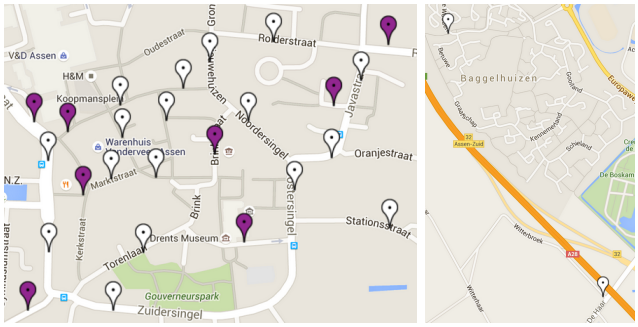


Fig. 1: Placement of scanners in the city center of Assen (left) and at the campsite near the city (right).

details on the type of noise one can expect; in Section V we present our methods for cleaning up this type of data sets; Finally in Section VI we present our results and conclude in Section VII.

II. Raw data set and Context

Trying to understand crowd dynamics requires timestamped localization data of the crowd. There are multiple ways to gather crowd localization data. However, in recent literature detecting WiFi-enabled devices (which are already ubiquitous given the rise in smartphone usage) using scanners is the method that stands out. This method is preferred because it is relatively cheap to deploy, nonintrusive, and requires little to no cooperation from the crowd that is being monitored. Details on the technology used to gather localization data in this manner and the inner-workings of a scanner are available in the literature [1].

The data set used in this paper was gathered in Assen, The Netherlands, during the TT festival [2] and in the few days after the festival (24-June-2015 to 30-June-2015). The festival lasted three days and consisted of 54 music events at eight different stages located in the city center. During the last day of the festival the TT NightRide was held, where a large group of motorcyclists followed a predetermined path cutting through the town center. On the 27th, the day after the festival the MotoGP [3] races took place at the Assen circuit. The festival was attended by an estimated 130,000 people.

The placement of the scanners can be observed in Figure 1: 25 scanners were placed in the city center, out of which eight near the music stages, marked with a darker color on the map; two of them were placed further from the city center, in the camping area and near the Assen circuit.

The platform of WiFi scanners consists of outdoor sensors (BlueMark 1000 series) and a server to store the results. For communication and data transfer, the 4G mobile network was used.

The Bluemark BM1000-sensor has 32 MB RAM, 8 MB flash and a CPU speed of 384 MHz. It runs openWRT as OS and an application that collects WiFi data. Moreover, it has a directional antenna with an antenna gain of around 12 dBi. The shape of the area in which WiFi packets can be received is irregular and inconsistent. However to improve visualizations, in this paper we will represent it as a 100m radius disc around the scanner. We emphasize that this visualization does not reflect in any way actual ranges, notably considering the use of directional antennas. More importantly we make no assumptions on the range in the solutions we present.

The sensor outputs data in SQL text format. This data is compressed and sent periodically to the server. At the server the output files are decompressed and set for long-term storage and analysis. For communication preferential SIMs are used along with a 4G dongle (Huawei E3276 LTE). In particular, the SIMs had higher priority over regular users of the network. During the event, this solution has proven to be robust. All scanners are synchronized using NTP and they reboot daily at 5am.

The scanner has been configured to detect only *Probe Request* messages. *Probe Requests* are sent by a WiFi device in order to search for available WiFi networks in the area. These packets are sent more or less periodically, even when a device is already connected to a network, in order to provide functions such as roaming. Smartphones also use them as a low-power and energy-efficient localization method [4].

Probe Requests contain the MAC address of the device. Because of this address we can uniquely identify a device across multiple scanners. In order to enhance privacy our system stores only hashed values of the MAC address along with the first 24 bits, representing the "Organizationally unique identifier" (OUI), which is used to identify the manufacturer of a device.

We use the following definitions:

- Device - A system with a WiFi module. Usually a smartphone or laptop, but it can also be, for example, a wireless printer or a router. The device is identified by its hashed address. These systems send WiFi packets.
- Scanner - A system that has a location, given by latitude and longitude, and an identifier. The system monitors WiFi frequencies and records all packets it receives, these are the packets sent by devices.
- Detection - A detection is a triplet that represents a device near the location of a scanner at a certain time. The triplet is $\langle \mathsf{scanner}, \mathsf{device}, \mathsf{time} \rangle$. A detection also has a received signal strength indicator value (RSSI) and number of packets that were received in that second.
- Movement Path - A set of consecutive detections of the same device.

The original data set contains 15,135,611 detections of 248,192 devices and spans over a 13-day period. In the following sections we discuss its transformation to a data set we believe is useful for further analytics.

III. Related Work

There is a lot of interest in understanding crowds and their movements. This is given by the large number of possible applications that can benefit from this information, applications such as measuring the size of queues of people [5] or disaster management [6]. As a consequence, numerous projects and techniques try to find ways of extracting this type of data.

Classically crowd data and crowd-location data is obtained using visual systems [7]. These systems are known to have a lot of noise because they require object recognition techniques which are still far from perfect. Noise is not absent from other methods such as GPS [8] or satellite systems like Argos [9].

In [10] the authors present a way of smoothing the path taken by an individual, as given by raw GPS data. The examples they show present a data set where multiple consecutive detections move back and forth, circling the street the individual is on. This behavior is similar to the behavior we present in this article. Yet the technologies and methods used are different. To extract a clear path, the authors use outlier removal with interpolation followed by Viterbi matching. Similarly [11] use outlier removal and Gaussian kernel regression to smooth the paths shown by GPS data. Their methods are not directly applicable to our scenario. When using GPS the data set consists of a high number of positions (equivalent to our detections), with error margins of just a few meters. In contrast, we have low number of detections and a rough approximation (in the order of hundreds of meters) of the actual position.

Because of the heavy use of smartphones, systems based on detecting them have come to light. These usually scan for Bluetooth or WiFi packets.

The WiFi scanners look inside the packets and extract information such as addresses [12]. With this information tracking crowds across multiple locations is possible. For instance in [13] the authors present a large-scale WiFi monitoring system used to gather information about facility planning, in their case a large hospital. The work goes in great detail on how the data is gathered and how information is extracted out of it. All this is similar to how data was collected in our scenario. The authors have encountered the same type of problems in the data as the ones we focus on in this paper. For example, their figures show a comparison between the real path taken by a device and the path that is extracted from the data. In [14] the same authors extend the previous work and offer a method for cleaning the data, proposing to just ignore most detections and keep the ones that have a large enough time difference between them. This method is a simplification of the time compression method we present in section V. We go much further in cleaning up and improving the data set. Similarly in [15] a time-based approach is taken in order to filter out devices that are static, seen by one scanner for a long time period.

It is possible to clear some noise through fine control of the hardware [16]. This however does not solve all types of noise and is dependent on other conditions.

As to our knowledge there is no previous work that directly tackles and solves the problem of noisy crowd-tracking data.

IV. Difficulties in data processing (Noise)

A perfect data set would be one in which the location of a device is accurately known at all times. This means that there is no time period in which there is no data about said device, and that when there is data, it is simple to pinpoint the device to a singular physical location. This does not mean that a device shouldn't trigger detections at two or more scanners simultaneously, given they are close enough. Simultaneous detections are acceptable as long as the RSSI values can be used to calculate a realistic positioning of the device. Obviously, perfect data sets do not exist for several reasons.

When trying to track crowds using WiFi, the data that needs to be analyzed is affected by errors from multiple sources. We consider the following ones.

a) Faulty scanner: Some are errors caused by the scanners and these are probably the simplest to detect and correct. For instance any interval in which a scanner is shut down or cannot receive packets will generate a clear irregularity in the density of detections over time for that scanner. In our example data set, scanners did an automatic reboot once every 24 hours, leading to a noticeable glitch in the detections.

b) Limitations of radio-based detections: WiFi uses a data transmission medium which is inherently unreliable [17]. For example, most WiFi devices claim a 100m transmission range in ideal conditions. In reality, such specifications cannot be relied on: while tunnels are generally known to extend the range, buildings and people are known to hinder transmissions. This also means that the shape or size of the area where WiFi packets can be correctly received can be very irregular. To illustrate, in our data set we have identified 1,491 occurrences when a device is detected by five or more scanners in the same second, yet the placement of the scanners was such that these simultaneous detections should normally not happen.

c) Limitations of RSSI: Using trilateration based on the received signal strength indicator (RSSI), we should, in principle, be able to pinpoint the location of a device. There are multiple problems to be addressed. First, the RSSI measurements as taken by the scanners, are not standardized and can differ in value or strength across different types of scanners. Second, the signal strength itself can dramatically differ across multiple device manufacturers and even different devices of the same model. Solutions for the RSSI problems are proposed [18] but only for when the mobile device is the one taking the measurements, and they do not directly apply to the reverse scenario. An experimental evaluation of RSSI-based localization methods is presented in [19], illustrating its inherent difficulties and certainly when applied to crowds.

d) Timing errors: Scanners timestamp detections. Consequently, their clocks may introduce many inaccurate detections if not properly synchronized between different scanners. One device moving from the range of scanner A to the one of scanner B could have a recorded detection at scanner B *followed* by a detection at scanner A. Even when the scanners are completely synchronized it may be difficult to determine the exact time a detection belongs to. There is no way to determine if two packets received at two different scanners are actually the same, as a *Probe Request* does not include a sequence number that would permit differentiating between two separate such packets from the same device.

e) MAC address issues: There used to be a time when a MAC address could be more or less used as a stable and

TABLE I: Sources of noise in set of detections

Description	Source			Correctable?
	Scanner	Device	Method	
Faulty scanner	Yes	No	No	Easy
Dynamic, irregular ranges	Yes	Yes	No	Hard
RSSI issues	Yes	No	Yes	Hard
Timing errors	Yes	Yes	Yes	Hard
Multiple addr. per device	No	Yes	No	Easy
Multiple devices per addr.	No	Yes	No	Easy
Uncoordinated probes	No	Yes	No	Medium
Lost packets	No	No	Yes	Medium

unique identifier for a device. This is no longer a justifiable assumption. Some devices change their MAC address seemingly at random, as also reported by [15]. This is known in particular in the case of some Apple devices [20]. Perhaps even worse when using a MAC address for device identification, is that we have noticed cases where different devices use the *same* MAC address.

f) Lack of coordination: Because there is no coordination between devices and scanners, no ideal probe transmission rate can be determined or let alone set. We have witnessed a huge variation in transmission rates, caused by seemingly random behavior concerning when a device switches its WiFi module on or off. This behavior is also dependent on the device, as reported in [1] where a comparison between Apple and Samsung devices is presented. As a result, the effect, in combination with the unreliability of the wireless medium [17] is a data set with detections that can make the movement of a device seem mostly erratic. To illustrate what this may lead to, consider Figure 2. In this case we have a known, nonmobile device appearing as a device that moves in circles, with random frequency and random speed. An actual mobile device could exhibit an even more chaotic behavior. Instead of moving in what would be a straight line, the detections would show it moving in small irregular circles while eventually getting closer to its destination.

By-and-large, there are many sources that introduce *noise* into a set of detections. Table I lists the main sources of errors that introduce seemingly chaotic behaviors in our data set. The last column notes the level of difficulty in dealing with this problems. The ones marked as hard could even be impossible to fix.

V. Methods of improving the data set

We now move to the actual data cleaning and separate two distinct phases. First, during the basic data-cleaning phase we get rid of obviously erroneous data. In our case, these consist of nontraceable devices, data caused by known misconfigured devices, and multiple simultaneous detections. Secondly, and more interesting, we look at cleaning up the data in such a way that we can be more confident that the detections belong to mobile devices that are actually moving according to yet unknown patterns.

A. Basic data cleaning

In its original form the data needs to be processed using simple filtering and cleaning techniques [21]. These techniques are as follows.

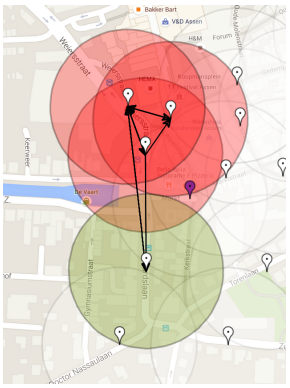


Fig. 2: Movement path of static device (the circles are 100m visual guides, they **do not** represent the cover radius)

a) Unkown manufacturers: We remove all data for devices that have unknown OUI values, as was also reported in [15]. To this end, we simply match OUI values to a publicly available list of manufacturers for network-enabled devices. We discovered that devices with unknown OUIs have very few detections, with an average of four per device. Furthermore, for these unknown OUI we have an average of only one device per OUI. All other devices have an average of 100 detections per device and on average there are 100 devices for every OUI. This is explained by devices that use randomly generated MAC addresses. We know this to be true for at least some Apple devices [20].

b) Misconfigured scanners: We also remove detections for the time periods where not all scanners were active or correctly placed. This happened in our case when the system was still being setup and tested. Note that deciding that the system is fully and correctly functional is not always easy to determine.

c) Simultaneous detections: There are situations where we have multiple detections for which the device, scanner, and time are the same. Such detections are simply merged to a single detection. Likewise, simultaneous detections, where a device is detected by multiple scanners at the same time, are filtered leaving just detections triggered at the scanner that has the highest RSSI score, as defined in equation 1. A similar problem and solution is described in [22], [23].

$$Score(\mathsf{S},\mathsf{d},\Delta T) = \sum_{\langle S,d,t\rangle, t\in\Delta T} RSSI * no\_packets \qquad (1)$$

Here:

- S represents a scanner
- d represents a device
- ΔT represents a time interval
- t represents a time value
- *RSSI* represents the signal strength of detection $\langle \mathsf{S},\mathsf{d},t\rangle$
- *no_packets* represents the number of packets that were received at detection $\langle \mathsf{S},\mathsf{d},t\rangle$ (in order to preserve bandwidth multiple packets can be merged into one detection)

After the data cleanup we have 10,673,498 detections of 127,959 unique devices spanning over six days. We consider this to be the *basic data set*, which we use in all our experiments. The techniques presented so far are necessary but not sufficicient, in the sense that they do not provide a data set where the movement path of a crowd can be easily

identified. This is made evident in Figure 2 where we map the movement path of a device from a WiFi router manufacturer that we discovered in our data set. Green represents detections with low RSSI and red represents strong detections, with high RSSI. This device is most likely a static router that is detected by multiple scanners. To draw the path we take all detections in a 10-minute interval and draw arrows between scanners, in increasing time order of the detections. We remind the reader that these are not simultaneous detections, rendering this behavior of the static device chaotic. We found this type of erratic behavior to be the norm, rather than the exception. The behavior is not particular to our data set, as we encountered it in other works, like the visualizations in [13]. It is also present in the case of GPS localization, as described in [24].

B. Advanced data cleaning

A mobility data set of a crowd needs to be modified in order to better permit information retrieval, such as correctly differentiating between mobile and static devices. Next we present three methods that aim to achieve this goal. The methods we present in this paper are executed on the basic data set.

The three methods, or rules, represent ways of modifying the data set, in which we smoothen the path each device takes through the city. This means obtaining a data set in which we minimize the device's behavior of moving in circles or constantly going back and forth between a set of scanners. To be able to measure the success of these methods we define two metrics.

First, we use **entropy**: taking two consecutive detections of one device, we calculate the probability of the second detection being at a specific scanner, given the first detection. The entropy is modeled as the Shannon entropy [25] as defined in equation 2.

$$H(\mathsf{S}) = -\sum_{S^*} p(\mathsf{S}^*|\mathsf{S}) \log \; p(\mathsf{S}^*|\mathsf{S}) \tag{2}$$

Here, p represents the probability that a device triggers a detection at scanner S^* right after a detection at scanner S. With 27 scanners, the entropy is calculated for each scanner and each entropy is calculated using 27 probability values. The average value of these entropies is used.

The second metric represents the **dissimilarity** between the new, computed data set and the basic data set. Given two data sets, with the same number of entries, and with matching values for device and time across them, we define the dissimilarity between the two data sets as the average Euclidean distance between the physical locations of the scanners of matching pairs of entries. For instance, if we only had one detection and we changed it from scanner A to scanner B the dissimilarity value would be the distance between scanners A and B. The more this value grows, the more information about paths taken is lost. Thus, if any *smoothing* of successive detections would eliminate paths taken by devices, this would result in a relatively high dissimilarity value. With our definition of dissimilarity, small values mean a high correlation between the two data sets and large values mean a low correlation. Note that the goal is not necessarily to reach a minimal dissimilarity value, this would mean little or no anomalies and no noise is removed from the paths.

a) Weak detection removal: One of the most obvious solutions for smoothing paths taken by devices represents the marking (for removal or change) of detections that have low RSSI values. We call this the *RSSI method*. The intuition is that such detections are of low quality, generated by scanners that are far away from the device and they were triggered by chance. This method is dependent on what we refer to as the *R threshold*, where any detections with RSSI smaller than *R* are marked as low quality. This is similar to the RSSI cleanup done in [26]. The authors mention that they have an RSSI threshold, similar to ours, but they do not discuss on how the threshold is chosen. Choosing the threshold is vital because there is no one-size-fits-all solution: bad weather conditions and placement of the scanners might require a very low threshold. We offer suggestions on how to choose the best threshold but a dynamic approach might prove to fit best.

b) Time compression: The next method consists in splitting the time in buckets of ΔT seconds, followed by identifying the sensor that has the highest strength score based on equation 1. Each bucket will then have only one detection per device. All others are marked for removal or change. Here we try to identify the main scanners at which a device triggers detections and leave only those in the data set. We call this the *time-compression method* because it lowers the resolution of the data set by moving from detections that can appear in any second, to only one detection per time interval of length ΔT. This method is related to the method used to clean up data in [14]. However, instead of making buckets and calculating the dominant scanner, the authors of [14] select detections that have a time interval ΔT between them. Without the selection of a dominant scanner, their method is biased to choose detections that keep abnormal behaviors, for instance keeping a detection of low quality (low RSSI) while removing one of high quality.

c) Cycle removal: Our last method is called *cycle-removal method* and it is specifically designed to remove the moving in circles, or going back and forth, phenomenon that we observed in the data set. To be able to remove this type of behavior from the data set we have to identify the intervals that display it. An interval with anomalous behavior has:

- consecutive detections of the same device
- the first and last detection at the same scanner
- repetitive detections at this scanner, with no more than *X* *detections* at other scanners between them.

X represents the maximal number of detections at other scanners, but that we would not consider to jointly form a circular walk. These detections are therefore marked as unacceptable. Any cycle that consists of more than *X* detections is considered a normal movement, a person literally moving in a circle. We use *X* as our parameter for tuning the removal of cycles.

Given the intervals we mark (for removal or change) all the detections that do not belong to the dominant scanner of

the interval, the one with the highest strength score, given by equation 1. Each detection can belong to multiple intervals. When faced with this we choose the interval with the earliest start time. This is done because we want to benefit the interval, and with it the cycle, that has the longest history.

Algorithm 1 shows the steps that need to be taken to identify and remove the cyclic movements using this method.

```
for each device do
    construct intervals
    for each interval do
        find scanner with highest score value (St)
    end
    for each detection do
        get first interval containing detection
        if scanner != interval dominant scanner then
            mark detection
        end
    end
end
```

Algorithm 1: Cycle removal

d) Applied modifications to the basic data set: Our methods for smoothing the paths can be applied in two different ways. One way is to simply remove all detections marked by it as unacceptable, we use the method as a filter. The other way is to change the scanner of these detections to one. That, according to the specific rule being used, is more appropriate.

For the RSSI method this means finding a detection of the same device close to the marked one (in the ordering of detections by time) that has an RSSI value above the threshold. We update the scanner of the marked detection to the scanner of this one. For the time-compression method this means changing the scanner value of all detections in the same time interval to that of the dominant one, according to *Score* from equation 1, for that interval. Finally, for the cycle-removal method this means changing the marked detections to the scanner that is the dominant one, according to *Score* from equation 1, in the interval the detection belongs to.

The entropy value can be calculated regardless if we remove or update detections, while the dissimilarity measure can be calculated only if we update them.

Both ways of applying the methods have their merits, the first allows us to better identify the points of interest when using different settings (i.e., for different values of R, ΔT or X, respectively). The second permits the use of the dissimilarity measure, because it results in data sets with the same number of data items as the base one.

VI. Experimental Results

We compare the three methods presented in Section V by running them on the basic data set. Each method has a variable that affects the aggressiveness of the data-cleanup process and we start by comparing the effectiveness of each method given different values for them. These variables are R, ΔT and X for tuning RSSI, time compression and cycle removal, respectively. We also present a direct comparison between the three methods and discuss the effects they have on the data set.

We remind the reader that we apply all three of our methods in two ways: we change our data set by either modifying or removing detections. When we modify detections we always get smaller values for entropy than when we remove detections. This is because by, for example, associating a different scanner with a specific detection to one that is dominant, we obtain a data set with more consecutive detections that have the same scanner. This in turn means a higher probability of two consecutive detections in the data set having the same scanner, which in turn leads to a lower entropy.

We first apply the RSSI method as intuitively changing or removing detections with low RSSI values should immediately lead to improving our data set. We have detections with RSSI values between -21 and -89. A value of -21 indicates an extremely strong detection, probably of a device that is right next to its detecting scanner and -89 represents a very weak detection. More than 95% detections have an RSSI value between -57 and -89. These are the RSSI values we use for tuning the R threshold. We run the algorithm creating a new data set for each threshold (in steps of 1 unit) and measure the entropy and dissimilarity for the new data set.

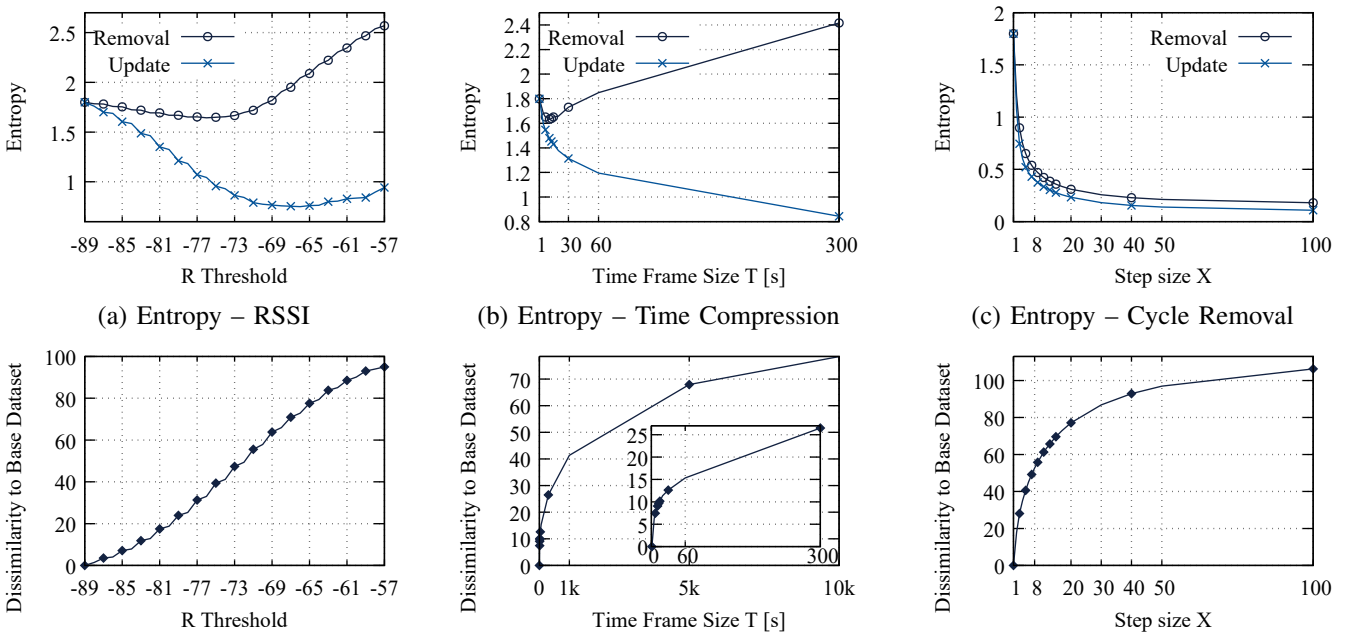
A. RSSI: Entropy

The entropy of the data sets thus obtained given the different values for R is shown in Figure 3(a). The lower the entropy, the more stable the system, the more predictable a following detection would be. A value of -89 represents the lowest threshold. Using it generates a data set identical to the basic data set. The entropy starts going down with an increase of R because with higher values of R we change or remove more detections that are of low quality, i.e. detections in which the device is actually far away from its detecting scanner. High values for R means we are accepting only high-quality detections, and the more we raise the threshold, increasingly fewer detections are available. This means that for the case of removing detections, we are starting to have more consecutive detections that are at scanners that are no longer close to each other, because the ones that were in between are detections of slightly worse quality and are marked for removal. This leads to an increase of entropy for relatively large values of R.

For very large R we encountered scenarios where no detections of a given device have an RSSI value greater than R. When we run the algorithm in order to change detections, we cannot find a better value to use and we leave all of them unchanged. This explains why with high values of R the entropy starts going up for the scenario of modifying detections. When we run the algorithm to remove detections, all of them are removed.

B. Time compression: Entropy

Figure 3(b) shows the entropy values for the time-compression method given different values for the time interval ΔT. When we remove detections the entropy goes down, with an increase of the time interval, and starts to grow only



(a) Entropy – RSSI (b) Entropy – Time Compression (c) Entropy – Cycle Removal

(d) Dissimilarity – RSSI (e) Dissimilarity – Time Compression (f) Dissimilarity – Cycle Removal method

Fig. 3: Entropy and dissimilarity values after using the RSSI, time-compression, and cycle-removal method.

when the time interval is too large. The reason this happens is because when the time interval is small we are leaving only detections that are the most important for that interval, removing those detections that taken together look and behave like noise. But when the time window grows too much we are starting to lose information. Consecutive detections are at scanners that are physically distant and the detections that were in between are lost. When we modify detections instead of removing them we have an identical phenomenon but it is hidden because we are left with many consecutive detections that share the same scanner. The more we increase the time interval, the more consecutive detections we see with the same scanner. When the interval is large enough, larger than the time in which the data set was collected, we are left with only one dominant scanner for each device. In that case, the entropy drops to 0.

C. Cycle removal: Entropy

Using the cycle-removal method we construct new data sets for different values of X. Recall that X stands for the number of tolerated intermediate detections at other scanners when measuring repetitive detections at one specific scanner. The entropy for these new data sets are presented in Figure 3(c). The entropy drops extremely fast and continues to slowly go down for higher values of X. Entropy never reaches 0, because not all our devices have repetitions, i.e., cycles in their detections. This is especially true for some devices that we see only once, entering the town center at one end, crossing it and exiting at another side, never to be seen again. Because this solution is targeted specifically at removing irregularities in the data, in none of the two cases does the entropy go up. The larger the step, the more irregularities are removed.

D. Dissimilarity measures

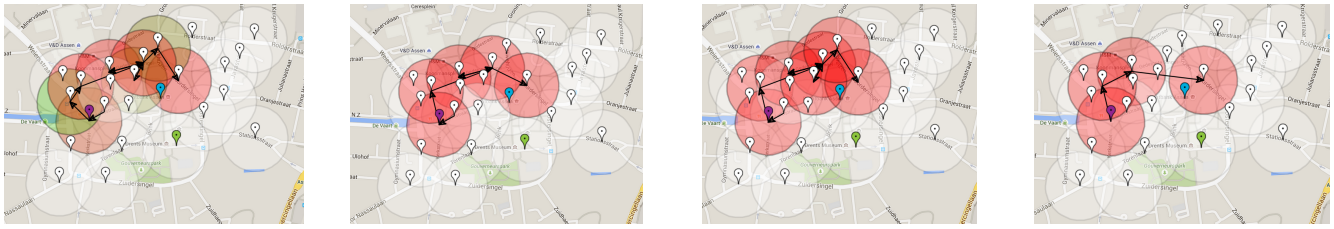
A drop in entropy by itself is not necessarily good, for instance with entropy of 0, we would have no information about how crowds move in the city. The lower the entropy the more information is lost, some information is just noise, which we want to remove and other information is data about the paths.

To understand how much information we lose when applying our algorithms we compared the resulted data set, obtained by changing the detections and keeping the same number of detections, with the basic data set. The metric we use is dissimilarity and the lower the value of this metric, the closer the new data set is to the basic date set. Having a dissimilarity value of 0, means we did not do anything and we did not remove any faulty behavior or noise. But having a dissimilarity value that is very large could mean we modified the data set too much and that we have lost or corrupted a lot of information.

As we can see in Figures 3(d), 3(e) and 3(f), respectively, the more aggressive we are about changing the data set the more the dissimilarity value rises, and thus the more the new data set is different from the original one. This is true regardless of method, however the speed with which it grows is different, time compression being the one where dissimilarity increases the slowest.

E. Comparing methods

To compare the results we select data sets generated by each algorithm, by choosing the best values for R, ΔT and



(a) Basic data set (b) RSSI, $R = -75$ (c) Time compression, $\Delta T = 11$ (d) Cycle removal, $X = 4$

Fig. 5: 10-minute path made by a device (the circles are 100m visual guides, they *do not* represent the coverage radius).

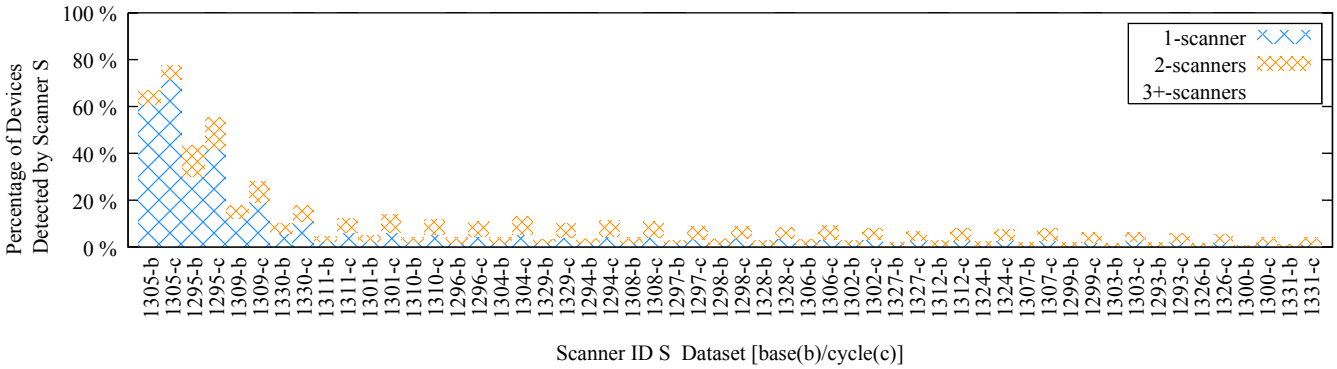


Fig. 6: Percentage of static and mobile devices seen by each scanner, before and after the cycle-removal algorithm with X equal to 4.

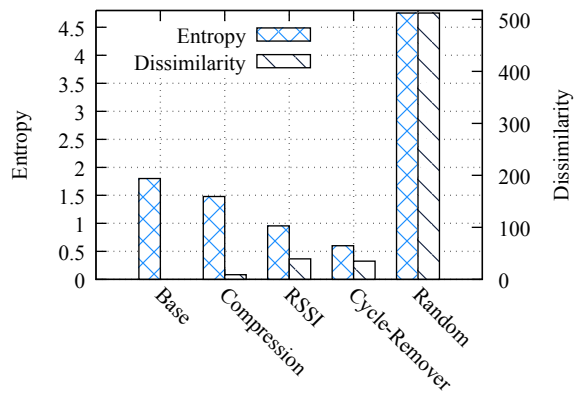


Fig. 4: Comparison between solutions, base and random

X, respectively. To choose these we take into account both entropy and dissimilarity. For the RSSI method we chose the data set with an R threshold of -75. This is the point corresponding to the lowest point of entropy in Figure 3(a), on the removal set. For the time-compression method we chose the time interval, ΔT, of 11 seconds, similar to the RSSI threshold as the point in Figure 3(b) where the entropy for removal is lowest. In the cycle-remover data sets the point with the lowest entropy also has the worse dissimilarity value, so instead we chose the point from Figure 3(c) where the rate of drop becomes almost constant, that is where X is 4.

For a better comparison we added the basic data set and a set obtained by changing the scanner in all detections to a random one, which we call the *random data set*. Having random scanners provides us a worst-case scenario. The maximal entropy is achieved when a detection is followed with the same probability, by a detection at any scanner. In our case this means a probability of $1/27$ for each scanner and an entropy of 4.75. In comparison, the entropy of our basic data set is 1.79. The minimal system entropy is 0. The latter can be achieved when each device is detected by only one scanner, in which case all devices are statically located. The dissimilarity is 0 when we compare a data set with itself. The results of these comparisons are shown in Figure 4 where we plot the entropy and dissimilarity values for these cases. There is no clear best method, one where both the entropy and dissimilarity are smallest compared to the others. Choosing the method and the appropriate value for R, ΔT and X, repsectively, is dependent on the application. Regardless, both entropy and dissimilarity need to be taken into account.

F. A path by a single device

Figure 5 shows the path a device has taken given each data set created with our methods, with the values of R, ΔT and X explored as just described. The path represents the same 10-minute window of a specific device. The colored scanners are the ones that detected the device. Low (bad) RSSI values are depicted with red and high (good) RSSI values are green. The arrows represent the direction the device took, given by the ordering of detections in time. From these figures it is

clear that all three methods improve the data set, but only the cycle-removal method offers a clear path that this device took. In Figures 5(b) and 5(c), most low-quality detections are removed but there are moments when the device seems to go against the normal direction.

G. Static versus mobile devices

Finally we were interested to discover how smoothing out the paths helps with identifying static devices. We used the same data sets with the values for R, ΔT and X as explained above. Figure 6 shows the percentage of devices detected by one, two or more scanners, respectively. Devices detected by one or two scanners can be considered static devices, the ones that are detected by three or more can be considered mobile. To have a better understanding the graph shows these values for all 27 scanners before and after the cycle-removal algorithm is run. We did not display the results for the RSSI method and time compression because they differ only slightly from the basic data set.

Some scanners detect a lot more static devices, for instance scanners with ID 1305 and 1295 (scanners from the camping sites placed away from the city center). These scanners detect a large number of devices carried by people that use the highway, positioned next to the scanners. These devices are usually detected only once. Scanners with IDs 1309 and 1330 are scanners at the Eastern edge of the city center, furthest away in the cluster of 25 scanners, with the smallest overlap with other scanners. This could explain why they have a higher number of static devices than the rest of the scanners in the city center.

Considering the low entropy, cleanest path and highest increase in the number of identified static devices we believe cycle removal with X equal to 4 to be the most appropriate method.

VII. Conclusion

WiFi tracking data taken as it is, shows abnormal behavior such as devices moving in circles or going back and forth. There are numerous sources for this type of noise in the data and each raises different problems. Use of this type of data directly can introduce a multitude of errors depending on the application. For instance a visualization tool for paths could display abnormal, chaotic movement; when trying to count the number of individuals in a flow, the results would be insignificant.

In order to clean the data we offered three distinct solutions that attack the problem from distinct points: One uses RSSI values; another uses time frames; and finally the most complex ones uses cycles, repetitions in the data set.

To validate these solutions we defined two metrics: the entropy which measures how predictable consecutive detections are, and the dissimilarity to the base data set measuring the average distance between the scanners in the two sets.

Using these metrics we show that each solution has its own advantages, and finding the best method is dependent on the application. However, in the cases of cleaning paths and identifying static devices, cycle removal performs best, it also has the lowest entropy, from the best cases chosen from all methods.

As future work we believe it is best to apply these methods on a data set that has ground truth. Testing different analytics applications on the cleaned data sets could give more indication as to what the best method is.

Acknowledgment

The work has been funded by the national project MobiWay, Project PN-II-PT-PCCA-2013-4-0321. We thank Roel Schiphorst from BlueMark Innovations for providing us with the Assen data set.

References

[1] M. Cunche, "I know your MAC address: Targeted tracking of individual using Wi-Fi," *Journal of Computer Virology and Hacking Techniques*, vol. 10, no. 4, pp. 219–227, 2014.

[2] TTFestival - http://www.ttfestival.nl/ (accessed: 2015-09-14).

[3] MotoGP - http://www.motogp.com/en/event/Netherlands (accessed: 2015-09-14).

[4] P. Sapiezynski, A. Stopczynski, R. Gatej, and S. Lehmann, "Tracking human mobility using wifi signals," *PLoS ONE*, vol. 10, 2015.

[5] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, "Measuring human queues using wifi signals," *Proceedings of the 19th annual international conference on Mobile computing & networking*, pp. 235–238, 2013.

[6] U. Walder, T. Bernoulli, and T. Wießflecker, "An indoor positioning system for improved action force command and disaster management," *Proceedings of the 6th International ISCRAM Conference*, pp. 251–262, 2009.

[7] M. Rodriguez, I. Laptev, J. Sivic, and J.-Y. Audibert, "Density-aware person detection and tracking in crowds," *IEEE International Conference on Computer Vision (ICCV)*, pp. 2423–2430, 2011.

[8] R. B. Langley, "GPS receiver system noise," *GPS world*, vol. 8, no. 6, pp. 40–45, 1997.

[9] I. D. Jonsen, J. M. Flemming, and R. A. Myers, "Robust state-space modeling of animal movement data," *Ecology*, vol. 86, no. 11, pp. 2874–2880, 2005.

[10] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones," *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pp. 85–98, 2009.

[11] Z. Yan, C. Parent, S. Spaccapietra, and D. Chakraborty, "A hybrid model and computing platform for spatio-semantic trajectories," *The Semantic Web: Research and Applications*, pp. 60–75, 2010.

[12] B. Bonne, A. Barzan, P. Quax, and W. Lamotte, "WiFiPi: Involuntary tracking of visitors at mass events," *IEEE 14th International Symposium and Workshops on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6, 2013.

[13] 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," *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 130–138, 2014.

[14] T. S. Prentow, A. J. Ruiz-Ruiz, H. Blunck, A. Stisen, and M. B. Kjærgaard, "Spatio-temporal facility utilization analysis from exhaustive wifi monitoring," *Pervasive and Mobile Computing*, vol. 16, pp. 305–316, 2015.

[15] A. Musa and J. Eriksson, "Tracking unmodified smartphones using wi-fi monitors," *Proceedings of the 10th ACM conference on embedded network sensor systems*, pp. 281–294, 2012.

[16] N. Abedi, A. Bhaskar, and E. Chung, "Bluetooth and wi-fi mac address based crowd data collection and monitoring: benefits, challenges and enhancement," *36th Australasian Transport Research Forum (ATRF), Brisbane, Queensland, Australia*, 2013.

[17] D. C. Salyers, A. D. Striegel, and C. Poellabauer, "Wireless reliability: Rethinking 802.11 packet loss," *International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–4, 2008.

[18] Y. Kim, H. Shin, and H. Cha, "Smartphone-based wi-fi pedestrian-tracking system tolerating the rss variance problem," *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 11–19, 2012.

[19] G. Zanca, F. Zorzi, A. Zanella, and M. Zorzi, "Experimental Comparison of RSSI-based Localization Algorithms for Indoor Wireless Sensor Networks," *Proceedings of the workshop on Real-world wireless sensor networks*, pp. 1–5, 2008.

[20] D. Stites and K. Skinner, "User privacy on iOS and OS X," *presented in Session 715 of Core OS WWDC14*, 2014.

[21] C. Chilipirea, A.-C. Petre, C. Dobre, and M. van Steen, "Filters for Wi-Fi generated crowd movement data," *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 285–290, 2015.

[22] E. Kalogianni, R. Šilerytė, M. Lam, K. Zhou, M. van der Ham, E. Verbree, and S. van der Spek, "Passive wifi monitoring of the rhythm of the campus," *AGILE*, 2015.

[23] W. Qin, J. Zhang, B. Li, and L. Sun, "Discovering human presence activities with smartphones using nonintrusive wi-fi sniffer sensors: The big data prospective." *International Journal of Distributed Sensor Networks*, p. 12, 2013.

[24] N. J. DeCesare, J. R. Squires, and J. A. Kolbe, "Effect of forest canopy on gps-based movement data," *Wildlife Society Bulletin*, vol. 33, no. 3, pp. 935–941, 2005.

[25] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

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