2016 17th IEEE International Conference on Mobile Data Management

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*—Crowd Monitoring is receiving much attention. An increasingly popular technique is to scan for mobile devices, notably smartphones. We take a look at scanning for such devices by recording WiFi packets. 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 the various techniques. 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. This leaves the experts to rely mostly only on their experience when it comes to preparing for an event. 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: one places WiFi scanners akin to normal access points, capable of sniffing network addresses. A level of privacy is added by hashing these addresses. The result is a data set consisting, conceptually, of timestamped (scanner, device) pairs. This should allow us to discover movements as they take place in a crowd.

We recently conducted an experiment with such a setup (Section II). 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 multiple and various noise sources (Section IV).

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 in such a way that seemingly erratic and senseless detections are transformed or eliminated. 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 the proposal of three distinct rules that improve the data set looking at: low-quality detections, averaging detections over a time period and, eliminating repetitive behavior in the form of unlikely cycles in movement (Section V). We also define two metrics aimed at determining the effect of our methods: (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. Finally we present the results (Section VI).

II. Raw data set and Context

Trying to understand crowd dynamics requires timestamped localization data of the crowd. Out of the multiple ways to gather crowd localization data, 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 relatively cheap to deploy, nonintrusive, and requires little to no cooperation from the crowd that is being monitored. Details on the technology are available in the literature [1].

The data set used in this paper was gathered in Assen, The Netherlands, during the TT festival [1] 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. The festival was attended by an estimated 130,000 people. The placement of the scanners can be observed in Figure 1. Eight out of 27 are placed near the music stages, marked with a darker color on the map.

[1] http://www.ttfestival.nl/ (Accessed: 2015-09-14)

2375-0324/16 $31.00 © 2016 IEEE
DOI 10.1109/MDM.2016.42

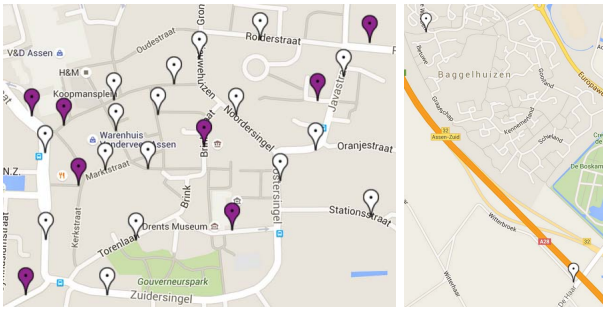


Fig. 1: Placement of scanners in the city center of Assen (left) and at the campsite near the city (right).

The WiFi scanners are from BlueMark 1000 series. They have 32 MB RAM, 8 MB flash, a CPU speed of 384 MHz, run openWRT and use 4G for communication. They use a directional antenna with a gain of around 12 dBi. The scanners connect to a central server that gathers and stores the data. They are configured to detect only *Probe Request* messages. *Probe Requests* are sent by most WiFi devices in order to search for available WiFi networks. These packets are sent more or less periodically, even when a device is already connected to a network, in order to provide roaming functionality. They contain the MAC address of the device. For privacy, the system stores only hashed values of the address along with the first 24 bits, representing the "Organizationally unique identifier" (OUI), which is used to identify the manufacturer of a device.

The original data set contains 15,135,611 detections of 248,192 devices and spans over a 13-day period. We filter the data using simple filtering and cleaning techniques, as described in our previous work [2]. A few examples of these filters are the removal of unknown manufacturers, based on OUI, the removal of time periods when the scanners were misconfigured (such as the testing phase) and removing multiple copies of the same 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 *base data set*, which we use in all our experiments. These filtering techniques are necessary but not sufficient.

III. Related Work

Considering the importance of crowd mobility and its many applications there are few works that deal with cleaning this type of data. In [3] 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 [4] 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.

Looking specifically for applications using WiFi scanners, such as [5], where the scanners are used to track visitors at mass events or [6], where a large-scale WiFi monitoring system is used to gather information on facility planning, in their case a large hospital, we noticed that other researchers encountered problems with noise. For instance [6] shows a comparison between the real path taken by a device and the path that is extracted from the data, exhibiting large deviations. Their method for cleaning the data is 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 [7] a time-based approach is taken in order to filter out devices that are static, seen by one scanner for a long time period.

We found multiple mentions of noise or cleaning data, but, as to our knowledge there is no previous work that directly addresses the problem. In most cases it takes the form of a side note.

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

a) Faulty scanner: For instance, any interval in which a scanner is shut down or cannot receive packets will generate an irregularity in the density of detections over time. In our example data set, scanners did an automatic reboot once every 24 hours.

b) Limitations of radio-based detections: WiFi uses a data transmission medium which is inherently unreliable [8]. 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 not occur.

c) Limitations of RSSI: The RSSI measurements are not standardized and can differ in value or strength across different types of scanners or devices. 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 [9] but only for when the

mobile device is the one taking the measurements, and they do not directly apply to the reverse scenario.

d) Timing errors: Scanners timestamp detections. Consequently, their clocks may introduce many inaccurate detections if not properly synchronized between different scanners. Even when the scanners are completely synchronized it may be difficult to determine the exact time a detection has. There is no way to determine if two packets received at two different scanners are actually the same. A *Probe Request* does not include a sequence number.

e) MAC address issues: Some devices change their MAC address seemingly at random, as also reported by [7]. This is known in particular in the case of some Apple devices [2]. 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.

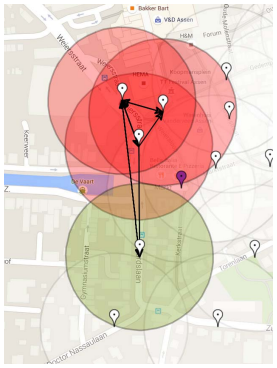


Fig. 2: Movement path of static device (the circles are 100m visual guides, they **do not** represent the cover radius)

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]. As a result, the effect, in combination with the unreliability of the wireless medium [8] is a data set with detections that can make the movement of a device seem mostly erratic.

By-and-large, there are many sources that introduce *noise*. Consider Figure 2. It was obtained by tracing the path a device took in a 10-minute interval. Green represents detections with low RSSI and red represents strong detections, while the arrows show the order in which the device was detected at different scanners. In this instance there are no simultaneous detections. This result is especially surprising considering the device is actually a WiFi Router (based on the OUI) and is most likely static. It is a non-mobile device appearing as a device that moves in circles, with random frequency and random speed. 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 [6]. It is also present in the case of GPS localization, as described in [10]. Things are worse for mobile devices which 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.

[2]User Privacy on iOS and OS X - in Session 715 of Core OS WWDC14

V. Methods of improving the data set

We showed in Section II that even after applying basic filtering techniques information retrieval from a mobility data set is not trivial. Even simple tasks such as differentiating between mobile and static devices requires expert intervention. This is caused by the high variety and intensity in the noise sources, as presented in Section IV. We define three rules, which when applied, aim to improve the data set.

The three 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 applying these rules we define two metrics.

The first metric is **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 [11] as defined in equation 1.

$$H(\mathsf{S}) = -\sum_{S^*} p(\mathsf{S}^*|\mathsf{S}) \log\ p(\mathsf{S}^*|\mathsf{S}) \tag{1}$$

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 base 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 from the two data sets. 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 update) of detections that have low RSSI values. We call this the *RSSI rule*. The intuition is that such detections are of low quality, generated by scanners that are far away from the device. This rule 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 [12]. 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 lower-than-usual threshold.

b) Time compression: The next rule consists in splitting the time in non-overlapping intervals of ΔT seconds and choosing only one scanner per device for each interval. The scanner is chosen as the one with the highest strength score based on equation 2. All detections not belonging to the chosen scanner are marked for removal or update. We call this the *Time Compression rule* because it lowers the temporal resolution of the data set. This rule is similar to the one used to clean up data in [6]. The authors of [6] 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 preserve abnormal behaviors. As to our knowledge we are the first to present a comparison between the usage of multiple values for ΔT or R.

$$Score(\mathsf{S},\mathsf{d},\Delta T) = \sum_{\langle S,d,t\rangle, t\in\Delta T} RSSI * no\_packets \qquad (2)$$

S represents a scanner, d a device, ΔT a time interval, t a time value, *RSSI* represents the signal strength of detection $\langle \mathsf{S},\mathsf{d},t\rangle$, finally *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).

c) Cycle removal: Our last rule is called *Cycle Removal* 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.

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 Cycle Removal rule.

Given the intervals we mark (for removal or update) all the detections that do not belong to the dominant scanner of the interval, the one with the highest strength score, given by equation 2. 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.

d) Applying modifications to the base data set: Our rules for smoothing the paths can be applied in two different ways. One way is to simply remove all marked detections, basically a filter. The other way is to change the scanner of these detections to one, according to the specific rule being used. We use both because even though the entropy can be calculated regardless if we remove or update detections, the dissimilarity measure can be calculated only if we update them.

```
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

For the RSSI rule this means changing it to the scanner of a detection of the same device, close in time to the marked one, that has an RSSI value above the threshold. For time-compression, 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 2. Finally, for cycle-removal this means changing the marked detections to the scanner that is the dominant, according to *Score* from equation 2, in the interval the detection belongs to.

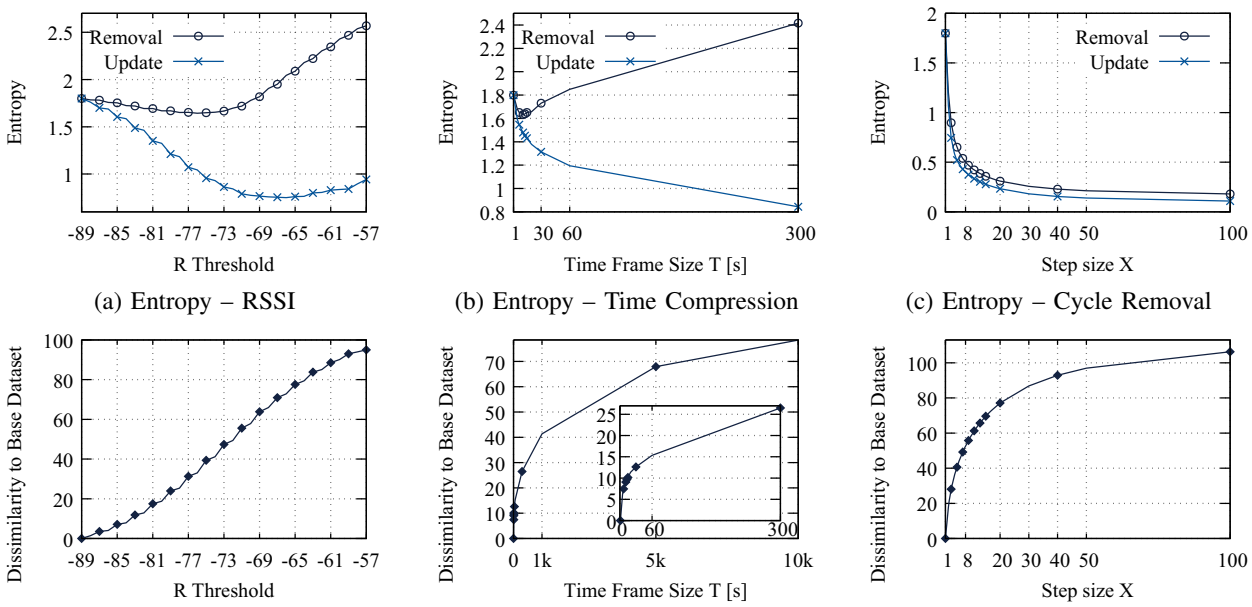
VI. Experimental Results

In order to properly understand the benefits and fall-offs of each of the rules we compare them using the data set presented in Section II. Each rule has a variable that affects the aggressiveness of the data-cleanup process. We compare the effectiveness of each method using different values for R, ΔT and X respectively.

We remind the reader that we apply all three of our methods in two ways, by either updating or removing detections. When we update detections we always get smaller values for entropy than when we remove detections. This is because by updating, we create many consecutive detections with the same scanner. We create two new data sets (one by removing detections and one by updating them) for each of the rules and each of the values for their control variables. Then we measure the entropy and dissimilarity of the new data sets. Like we mentioned in the previous section the smaller the values for both entropy and dissimilarity, the better.

The data set contains detections with RSSI values between -21 (strong) and -89 (weak). More than 95% detections have an RSSI value between -57 and -89. These are the values for which we set R. ΔT is measured in seconds and takes values from 1 to infinity. Finally, X can take any unsigned integer as a value (usually works best with small values 2-10).

a) Entropy: Figures 3(a), 3(b) and 3(c) show the resulting entropy for each of the three rules. The entropy starts going down when we increase the values of R, ΔT and X. In the case of RSSI it goes naturally that when we remove a small number of low quality detections, we remove noise and



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

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

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

the entropy decreases, when we increase the threshold, we remove more noise. Similarly, for Time Compression when we consider small intervals noise is removed. Finally, in the case of Cycle Removal when we increase X we remove larger cycles, larger noise parts of our data set. For Cycle Removal the entropy keeps going down because with large enough X we remove all cyclic behavior from our data set and leave only data for devices that never have a cycle, such as people passing through the city, and only few scanners for the ones that contain cycles.

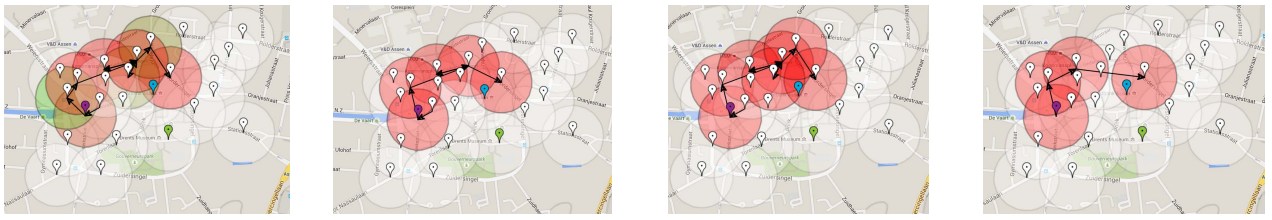
However, in the case of RSSI and Time compression the entropy starts increasing after a point. High values for R means we are accepting only strong detections, and the more we raise the threshold, increasingly fewer detections are available. This means that we are starting to have more consecutive detections that are at scanners which are no longer close to each other. A similar thing happens with Time Compression. When we increase the interval too much, the entropy increases because consecutive detections are frequently at scanners which are too far. The same thing does happen when we update the detections instead of removing them but the entropy continues to go down because we create more and more consecutive detections at the same scanner. The effect is not as apparent for RSSI because for a large enough R we encountered scenarios where no detections of a given device have a high enough RSSI value to pass the threshold and we leave those unchanged.

b) Dissimilarity measures: A drop in entropy by itself is not necessarily good. The lower the entropy the more information is lost. To understand how much data is lost when applying our rules, we compared the resulted data sets, obtained by updating, with the basic data set. Having a dissimilarity value that is very large could mean we modified the data set too much and we lost 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 comes naturally and is true regardless of method.

c) Comparing methods: To compare the results, we select a data set generated by each algorithm, by choosing the best values for R, ΔT and 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. Applying the same method for Time Compression we chose the time interval, ΔT, of 11 seconds. In the Cycle Removal data sets the point with the lowest entropy also has the worst 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. We also compare to a randomly generated data set. The results can be observed in Figure 5. We note that the entropy can be 0 if there is no movement data and the dissimilarity is 0 when comparing a data set with itself.

d) Mapping the path took by a device: Figure 4 shows the path a device has taken and how the path looks before and after applying the rules. We use the values of R, ΔT and X described previously. The path represents the same 10-minute window. The colors and arrow have the same meaning as the



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

Fig. 4: 10-minute path made by a device (the circles are 100m visual guides, they **do not** represent the coverage radius).

ones in Figure 2. These figures show that all three methods can improve the data set, but only the Cycle Removal method offers a clear path that this device took. In Figures 4(b) and 4(c), some noise is removed but there are moments when the device seems to go against the natural direction.

e) Static versus mobile devices: Using the same four data sets we compared the number of identified static devices. While in the first two cases there was not much difference in the percentage of static-to-mobile devices, when we applied the Cycle Removal rule we had between 3% and 10% more static devices. Devices such as the one in Figure 2 are now correctly classified as static.

Considering the low entropy, cleanest path and highest increase in the number of identified static devices we believe Cycle Removal to be the most appropriate for general use.

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. We showed that even simple tasks such as visualizing a path or differentiating static and mobile devices can by difficult without proper data manipulation.

In order to clean the data, we offered three distinct solutions: One uses RSSI values; another uses time intervals; and finally the most complex one 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 showed that each solution has its own advantages, and finding the best method is dependent on the application. However, cycle removal seems to be the most promising. It has the lowest entropy, and shows promising results when visualizing paths.

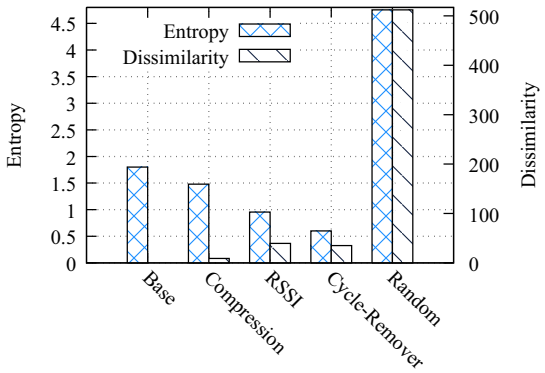


Fig. 5: Comparison between solutions, base and random

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.