Counting and locating people in outdoor environments: a comparative experimental study using WiFi-based passive methods

Antonio Guillen-Perez*, and *Maria-Dolores* Cano

Department of Information Technologies and Communications, Universidad Politécnica de Cartagena, 30202 Cartagena, Spain

Abstract. WiFi-based passive methods are becoming a common tool to count, estimate, and/or locate people. One area of applicability is the development of intelligent control system for traffic management in urban areas, so that these systems are able to take into account not only vehicles' behaviors but also pedestrians', as important actors in the road scenario. In this work, we present the performance evaluation in terms of accuracy of a WiFi-based passive method used to identify pedestrians, classify them as moving pedestrians or static pedestrians, and for the latter, to locate them in a traffic intersection. The proposed algorithm is implemented in a low-cost development board and tested through several experiments in a real outdoor scenario. Our proposal is compared with several classic Machine Learning (ML) algorithms, specifically with Binary Logistic Regression, Support Vector Classification, Gaussian Naive Bayes, Random Forest, and *k*-Nearest Neighbors. Results show that despite the simplicity of our method, the outcomes are similar or better than most of the ML techniques, without the expected complexity or computational requirements that the latter required.

1 Introduction

Most recent statistics confirm a continuous increase in the number of cellular phones worldwide. Based on data from the International Telecommunication Union (ITU), the World Bank has recently shown that the average number of mobile cellular subscriptions per 100 people is 104.49 [1]. In addition to all the benefits of having a cellular device, many industry sectors rather than the telecommunications companies could take advantage of this fact not only to improve current services and application but to offer new ones. For instance, assuming that each person carries a cellular phone with WiFi connectivity, we can estimate the number of people in a particular location or locate people in a particular place. The underlying idea is that mobile devices with WiFi connectivity send data frames, which can be captured and analyzed to obtain that information. Two different approaches can be followed to do so, namely, passive or active. WiFi-based passive methods do not need an established connection between mobile devices and Access Points (AP). On the other hand, active methods do require an established connection between wireless devices and AP in order to obtain the mentioned data.

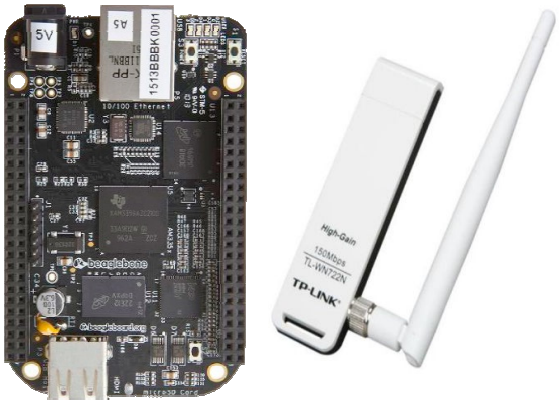
In a previous work [2], we proposed a WiFi-based passive method to estimate the number of pedestrians in urban outdoor scenarios and locate them. Our goal was to obtain these data so that intelligent systems developed to control signaled traffic intersections can react to the current traffic scenario taking into not only vehicle traffic but also pedestrians [3]. Pedestrians are also actors in the traffic urban scenario, but their behavior is not commonly included in elaborated solutions to avoid traffic congestion and improve the performance of urban traffic. In our opinion, with the advent of autonomous vehicles this paradigm should shift. Therefore, we designed our algorithm with that aim. In short, our method captures the WiFi Probe Request messages sent passively and autonomously by pedestrians' mobile phones. Mobile phones send Probe Request messages to find available WiFi networks to connect to. Once these captures are anonymized, the system was able to differentiate between pedestrians walking and pedestrians waiting to cross (i.e., waiting for the green signal to cross through the pedestrians crossing). In addition, it was able to locate the position where pedestrians were waiting.

Given the promising results that we obtained in the performance study of our proposal, via computer simulation, we decided to extend our work with real experiments. In this paper, we present the results obtained in a real outdoor environment using our WiFi-based passive method. We implemented the proposed algorithm in a Beaglebone Black, under Linux 7.8 (Wheezy), incorporating a WiFi USB adapter (TP-Link TL-WN722N). The designed software works under python 3.6, with specialized scientific computing libraries such as NumPy [4], SciPy [5] and pandas [6]. The devices used are shown below in Fig. **1**. In addition, we compare the performance of our proposal with

* Corresponding author: antonio.guillen@edu.upct.es

several Machine Learning (ML) algorithms, namely, Binary Logistic Regression, Support Vector Classification, Gaussian Naive Bayes, Random Forest, and *k*-Nearest Neighbors. The goal of this comparison is to identify if the simplicity of our proposal implies a trade-off in accuracy with other techniques that (initially) would have higher computation requirements. Note that the terms mobile device, cellular device, and mobile phone are used with the same meaning along this paper.

The rest of the paper is organized as follows. In Section 2 a brief state of the art is presented. Section 3 describes the experimental testbed. A description of the algorithms used is given in Section 4 and in Section 5 the accuracy of the described algorithms is obtained. Finally, the article concludes in Section 6 with the conclusions and future work.



(a) (b)

Fig. 1. Devices used in the experimental tests: (a) Beaglebone Black; (b) WiFi USB adapter TP-Link TL WN722N.

2 Related Work

One of the simplest methods to estimate the number of mobile devices, to track them, or to obtain a density estimation are WiFi-based passive methods. These methods are based on capturing messages that mobile devices send autonomously to discover WiFi networks to connect to. These messages are usually Probe Request messages, although others could be used. Despite the simplicity, the vast majority of works from the related literature study WiFi-based passive methods in indoor environments [7]–[13]. There is only a few of works focusing on outdoor environments. In this section we will briefly review these works, the outdoor experiments carried out, and the results that other authors have obtained.

In [14] the authors presented an interesting work about how pedestrian behavior could be measured employing passive WiFi algorithms. Specifically, they focused on studying a WiFi tracking system to analyze the behavior of pedestrians in open environments by capturing Probe Request messages and differentiating pedestrians simply by identifying the amount of different anonymized Medium Access Control (MAC) addresses included in those messages. After a filtering of the captured data the following information was eliminated: the duplicated data, the data captured outside the desired time-interval, messages with random MACs, and the messages coming from laptops. Then, the algorithm that they proposed simply counted the number of different IDs (OUI and MD5 hash of the MAC address) remaining at each time interval. Another interesting experiment that they carried out is the study of the period of sending Probe Request messages on all the devices they captured. That is, for each device captured with a real MAC, they measured the sending period of two consecutive Probe Requests and counted the number of occurrences. Although the highest value they obtained (of 3 seconds) may be due to one of the filters they proposed, other interesting values they obtained were around 30, 45, and 60 seconds. The evaluated scenarios were varied, e.g., complete urban areas and some social events. Although the authors did not offer results in terms of estimating the number of pedestrians, their results allow us to have an idea of how pedestrians move (pedestrians' flow) in all the mentioned cases studies and also gives us an idea of the frequency of sending Probe Request messages. These results can undoubtedly facilitate the work of event organizers, urban planners, etc. Similarly, the authors in [15] introduced another method analyzing pedestrians' flows using WiFi Probe Request messages sent by mobile devices carried by pedestrians. The system they proposed consisted of multiple low-cost distributed sensors within the area of interest, which captured the Probe Request packets sent by the mobiles, and then, this information was sent to a server in the cloud. This cloud storage server was responsible for calculating pedestrians' flows in real time. Messages sent to the cloud included the anonymized MAC, the sensor ID, a time stamp, and the received power. Once the cloud had the data of all sensors ordered by time and ID, it was easy to compute the flows. The results obtained from real experiments confirmed that pedestrians' flows can be approximately obtained with the proposed WiFi-based passive method.

Regarding the use of Probe Request messages to locate wireless devices in outdoor environments, Acuna *et al.* presented an interesting work [16]. In this work, the authors proposed the use of Unmanned Aerial Vehicles (UAV or drones) to locate people in cases of emergency. For this, a UAV with a GPS module flies over an area of interest, capturing the Probe Request messages sent by the mobiles, and by means of a Random Forest classification algorithm the approximate position of the mobile is obtained. The area of interest is divided into sectors of 7.5m x 7.5m and the results obtained in locating mobiles devices correctly within these sectors showed an accuracy of 81.8%. From a different perspective, Musa and Eriksson introduced in [17] a system for the flow estimation of mobile devices using Probe Request messages. They proposed a path estimation method based on the Viterbi algorithm that takes the detections of a mobile device in real time and produces the most probable route followed by each mobile device. For the study of the estimated path, they

proposed the use of WiFi detectors, separated between them more than 400 meters. In addition, WiFi sensors sent all data in real time to a central server, which was responsible for calculating the route. Detection and differentiation of devices was done through the MAC included in the Probe Request messages Although the results obtained showed a location error below 70 meters, it is worth mentioning that the devices for capturing these messages were more than 400 meters apart. Therefore, as they concluded, due to the low cost of deployment and implementation, WiFi-based passive methods are a great alternative to consider for large-scale deployments and very useful information can be easily obtained for many fields. Finally, in [18], a testbed was presented for the study of outdoor pedestrian flow mobility. For this, the author used 8 sniffers implemented in Beaglebone development boards powered by a solar panel. The differentiation between captured devices is done by means of the MAC included in the Probe Request captured messages. The processing was done in real time in the cloud and for this, the sniffers sent the captured data to the cloud using 3G. The results obtained show that valuable information on pedestrian mobility can be obtained easily. However, the main problem of the experiment was that authors had no control over the scenario, so the precision could not be measured.

Tree ideas can be derived from the related works in the scientific literature. First, the applicability of tools for counting/locating people is very wide. Second, the simplicity of WiFi-based passive methods, i.e., techniques based on the fact that pedestrians usually carry (at least) one wireless device with WiFi connectivity and these devices do not need to be connected to any AP in order to be detected, is also very high. Finally, despite the positive facts associate to these methods, more emphasis has been made in indoor scenarios than in outdoor ones, mainly due to the complexity of performing real experiments in controlled outdoor environments.

3 Experimental Setup

To test the accuracy of the proposed method we needed to know the real number of wireless devices. Therefore, experiments were carried out in a rural environment, where the influence of external wireless devices not belonging to our experiments could be avoided. This real scenario is shown in Fig. **2**. At each urban intersection we assume there are four Data Acquisition Units (DAU), basically a development board with a wireless interface able to capture WiFi data frames. Therefore, in our experimental deployment we used four DAUs. The distance between DAUs is 30 meters and they were located at a height of 2 meters. The scenario emulates the signaled traffic intersection represented in Fig. **3**. At the top of each traffic light in Fig. **3** there is a DAU that acts as a WiFi AP capturing probe request messages. Pedestrians either wait to cross or cross through the pedestrians crossing.

A total amount of 14 mobile phones were used in the experiments. The details of the mobile phones are described in Table 1. Note that we assumed that each pedestrian carries one cellular phone. Mobiles were in a state of passive search for WiFi networks, i.e., with the WiFi in an on-state without being connected to any network and the screen off.

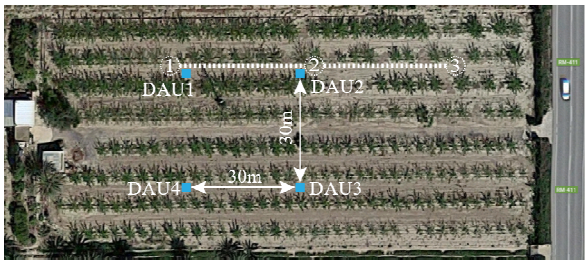


Fig. 2. Real outdoor scenario where experiments were performed.

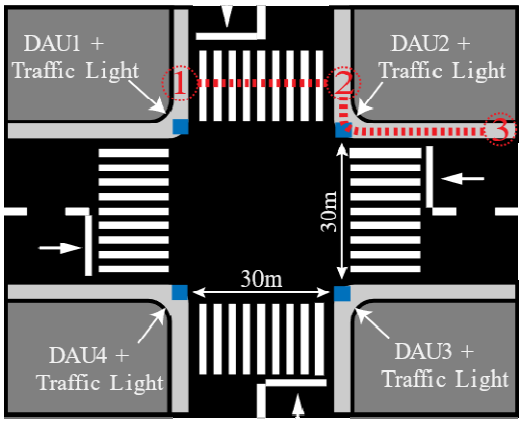


Fig. 3. A signaled traffic intersection composed by for edges. At each edge there is a traffic light that incorporates a Data Acquisition unit (DAU).

Table 1. Devices used, brand, model and version of the operating system (OS).

Brand/Model	S.O Version
Xiaomi Redmi Note 5	Android 8.1.0 MIUI 9.6
Samsung J5 2016 SM-J500F	Android 7.1.2
Samsung Galaxy S4 GT-I9506	Android 7.1.2
Samsung Galaxy S7 Edge	Android 7.0
Xiaomi Redmi Note 4	Android 6.0 MIUI 9.6
Huawei Y6 2017 (x2)	Android 6.0 EMUI 4.1
Vodafone Smart Turbo 7	Android 6.0
Xiaomi Redmi 3	Android 5.1.1 MIUI 9.6
Samsung Galaxy S2 GT-9100	Android 4.1.2

Google Nexus S	Android 4.1.2
Vodafone Smart III 975N (x2)	Android 4.1.1
Samsung Galaxy Ace GT-S5830	Android 2.3.3

For simplicity, all pedestrians followed the same movement pattern, which is represented as a dotted line in Fig. **2** and is equivalent to the dotted line in Fig. **3**. Experiments were divided into time intervals. At each time interval, pedestrians were either static or in motion. The speed at which pedestrians moved varied slightly around the average humans' walking speed (around 1.38 m/s). Thus, there were pedestrians that took a few seconds less to cross the intersection, and others that took a few seconds more.

Time intervals were set to 30 seconds, which can be considered the minimum time that a traffic light is in red state. Observe that the goal of our algorithm is to estimate the number of pedestrians waiting to cross a street and locate them in the intersection. Therefore, by using the minimum value of 30 seconds we will be in a worst-case scenario. In other works, this short duration allows us to capture the lowest number of probe request messages sent by mobile devices. In situations with larger traffic lights' phases more probe request messages could be obtain, consequently improving the accuracy of the algorithm (more information available better results, please see [2]). The pedestrians' movement pattern, the time intervals and the associate timestamp are detailed in Table 2.

All tested algorithms will be responsible for classifying pedestrians as moving or static (waiting to cross). If pedestrians are classified as static, the algorithm should also identify the closest DAU to those pedestrians. This location procedure, within the intersection, is performed by identifying the DAU with the highest average power level during the capture time interval. Please refer to [2] for a full definition of our proposal. The employed metrics to measure accuracy are defined as follows:

- *Amoving* indicates the accuracy of the classification algorithm to label pedestrians as moving and is calculated as indicated in equation (1). That is, *Amoving* is the ratio of pedestrians detected as moving by the algorithm over the real number of pedestrians who are walking.
- *Astatic* indicates the accuracy of the classification algorithm to label pedestrians as static and is calculated as indicated in equation (2).
- *Apositioning* indicates the accuracy obtained by the system in correctly positioning pedestrians who were detected as static ones. That is, it represents how many static pedestrians are properly located in the intersection (in any of the corners). This accuracy is calculated as shown in equation (3)

Table 2. Time intervals, timestamps, and pedestrians' state

Time Interval	Timestamp	State
0s-30s	T1	Static on (1)
30s-60s	T2	Static on (1)
60s-90s	T3	Static on (1)
90s-120s	T4	Static on (1)
120s-150s	T5	Moving from (1) to (2)
150s-180s	T6	Moving from (2) to (3)
180s-210s	T7	Moving from (3) to (2)
210s-240s	T8	Moving from (2) to (1)

$$A_{moving} = \frac{pedestrians\_detected\_as\_moving}{\text{total\_number\_of\_pedestrians\_moving}} \quad (1)$$

$$A_{static} = \frac{pedestrians\_detected\_as\_waiting}{\text{total\_number\_of\_pedestrians\_waiting}} \quad (2)$$

$$A_{positioning} = \frac{pedestrians\_located\_properly}{pedestrians\_detected\_as\_waiting} \quad (3)$$

We implemented the proposed algorithm in a Beaglebone Black, under Linux 7.8 (Wheezy), incorporating a WiFi USB adapter (TP-Link TL-WN722N) (see Figure 1). Regarding the software, we implemented and configured a sniffer that captures WiFi messages with the tcpdump tool, keeping only the Probe Request messages. From these messages, the following information is stored in an output file in csv format: the MAC of the DAU that has captured the message, the MD5 hash of the captured MAC, the time stamp of each message, and the captured power. To run the sniffer, we only need to put the interface in monitor mode (`iw dev wlan0 interface add mon0 type monitor`) and execute the following command:

```
tcpdump -l -I -i mon0 -e -s 256 -l type
mgt subtype probe-req -ttttt
```

In addition, in order to capture the maximum number of Probe Request messages, the sniffer changes channels cyclically every 2 seconds in all WiFi channels.

After capturing the messages in the desired time interval, the messages are sent to a master node within the intersection via another WiFi interface. When the master node has all the csv files, it uses a script developed in python 3.6 to perform all calculations. Because all calculations are done within the Beaglebone (a low consumption development board), the developed algorithm must be lightweight and highly efficient. Therefore high-performance libraries such as NumPy [4], SciPy [5], and pandas [6] were used to process all the data. In addition, ML algorithms are executed under the scikit-learn [19] library, which provides a large

number of traditional ML algorithms, such as classification, regression and clustering. In addition, it is designed to operate with high-performance scientific libraries such as NumPy and SciPy.

4 Description of the algorithms

In this section, we describe both our proposal to estimate and locate the number of pedestrians in an urban intersection and also some traditional ML algorithms that will be used for comparison purposes.

4.1 Our proposal

Our previously developed algorithm [2] allowed a quick pedestrian classification based on the power variance of the captured probe request messages. Briefly, a pedestrian whose mobile device presented in the captured data a low power variation was labeled as static, i.e., waiting to cross. On the contrary, if during the observed time interval, the captured data showed a high variation in the pedestrian's mobile device power, then our algorithm classified this pedestrian as moving. A more detail explanation follows:

- DAUs capture the probe requests messages sent by mobile devices during 30-second time intervals (worst case scenario), and MAC addresses are anonymized
- At the end of the time interval, all captured messages are sent to a predefined DAU, where the whole procedure that follows is performed
- Pedestrians are differentiated according to the anonymized MAC included in the probe request message
- For each pedestrian *p* that is identified, the DAUs that detected the pedestrian *p* are obtained, and they are stored in a two-dimensional array
- For each DAU in the array, the variance of the captured probe request messages is obtained
- If one of the DAUs shows a variance greater than a predefined threshold, the pedestrian *p* is considered as *moving*; otherwise, the pedestrian *p* is considered as *static*
- In addition, if only one probe request message is captured by a DAU during all the time interval, the pedestrian *p* is directly considered as *moving*
- Finally, the position of the *static* pedestrians is obtained by selecting the DAU with the highest measured average power of the captured probe request messages

4.2 Traditional Machine Learning Algorithms with same input variable

The traditional ML classification algorithms described in this section will be used only to label a pedestrian as *static* or as *moving*. The input data to these algorithms is the same as used by our proposal, that is, the variance of the probe request messages captured from each pedestrian in all the DAUs. Therefore, the performance of these ML algorithms will be compared with the performance of our proposal. However, the method to locate a *static* pedestrian will consist always on obtaining the DAU with the highest measured power disregarding the algorithm employed.

4.2.1 Binary Logistic Regression [20]

Binary logistic regression is an extension of simple linear regression. This method can model a binary variable by using a logistic function and a series of coefficients. The logistic function provides values between 0 and 1 with continuous inputs. The learning process allows adapting the coefficients that control the values of inputs and are calculated by means of a process called maximum-likelihood estimation.

4.2.2 Support Vector Classification [21]

The Support Vector Classification (SVC) algorithm belongs to the category of Support Vector Machines (SVM). SVC is a specialized variant for classification. Intuitively, the operation of SVM represents training points in space with as many dimensions as features have input data. After this, it separates the classes into two spaces as far apart as possible by means of a hyperplane. The closest points of this hyperplane are what are defined as "support vectors".

The classification is made according to the position in the mentioned plane of samples. The main advantage of SVM is its high efficiency in spaces of many dimensions (that is, when there is a large number of features) and its high versatility, being able to modify the kernel function among other parameters. However, if the number of characteristics is much higher than the number of samples, a kernel adjustment and adjustment in the regularization term is essential to avoid overfitting.

4.2.3 Gaussian Naive Bayes [22]

Based on the Bayes' theorem, the Gaussian Naive Bayes classifier allows for independent simple probabilistic classification. This classifier is (naively) based on the fact that there is no dependency between the variables. In other words, this classifier assumes that variables are not related to each other and the presence or absence of some variable does not imply the presence or absence of another. In the training process of a Gaussian Naive Bayes classifier, each sample is modelled as a Gaussian probability function and allows obtaining a classification based on a decision threshold. Commonly, this decision threshold is to choose the likeliest option; this is known as the maximum a posteriori (MAP) decision rule.

4.2.4 Random Forest Classifier [23]

Random forest classification algorithms are based on the combination of decision-tree classifiers: an ensemble of

decision trees in which each node is divided using the best subset of predictors. This combination creates a set of probabilistic decision rules that depends on each tree that makes up the set. The main objective with this approach is to reduce the variance while the bias remains the same. This approach allows to eliminate the overfitting that decision trees usually make by "averaging" among different decision trees, where each tree has been created with different decision rules.

4.2.5 kNN Classifier [24]

Differing from previous approaches, the *k*-Nearest Neighbors (*k*NN) classification method classifies new samples according to their nearest k-neighbors. That is, when locating a new sample, it looks at its neighborhood and classifies this sample as the more frequent class among neighbors.

5 Results

In this section we present and discuss the results obtained from the experimental tests. Not only the accuracy metrics introduced previously (i.e., *Amoving*, *Astatic*, *Apositioning*) will be shown but also the hyperparameters that maximize the outcomes for the ML methods.

First, we present the performance of our proposal. As it can be seen in Fig. **4**, the accuracy in differentiating between static and moving pedestrians is higher than 75% and 50%, respectively. In addition, the positioning accuracy presents values above 80%. These precisions obtained in the real outdoor experiments surpass those obtained in our previous work, which were based on computer simulations. This result can be due to the fact that the real communication channel presents fewer randomities than in the simulated channel in the simulator. Compared with other works from the related literature, our proposal presents a better performance. Note that to the authors' knowledge, there are no more works with the same approach of counting or locating pedestrians with the goal of improving the traffic scenario. Thus, our comparison is with other type of applications.

Second, we present the outcomes obtained using some traditional ML methods, namely, Binary Logistic Regression, SVC, Gaussian Naive Bayes, Random Forest, and k-NN. The results obtained using SVC are worse than those obtained by our proposal, being even lower than 50% for the moving accuracy. All values are shown in **Fig. 6**. In this case, the optimal hyperparameters for SVC are shown in Table 3. Given that the operating principle of Binary Logistic Regression is very similar to the one we suggest in our proposal, the results are also very similar, as depicted in Fig. **5**6. The optimal hyperparameters for this ML classification algorithm can be found in Table 4. Likewise, the outcomes achieved using the Gaussian Naive Bayes classifier are also similar to those obtained by our proposal (see Fig. **7**).

On the other hand, the results obtained using the Random Forest classifier are better than those obtained with the other algorithms. It presents an accuracy higher than 80% to identify moving pedestrians, and higher than 75% for static pedestrians, as represented in Fig. **8**. Over all pedestrians identified as static, more than 80% are properly located. The optimal hyperparameters for the Random Forest classifier are given in Table 5.

Finally, the results obtained for the *k*NN classifier are quite fair compared to our proposal, and similar to those obtained with the random forest algorithm (see Fig. **9**). The accuracies are maximized when the hyperparameters shown in Table 6 are selected.

Table 3. Optimal hyperparameters for SVC.

Hyper-parameter	Optimal Value	Brief description
C	0.01	Penalty parameter of the error term
class_weight	balanced	Weights associated with classes. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data
kernel	linear	Specifies the kernel type to be used in the algorithm.

Table 4. Optimal hyperparameters for Binary Logistic Regression.

Hyper-parameter	Optimal Value	Brief description
C	0.001	Inverse of regularization strength
class_weight	balanced	Weights associated with classes. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data
penalty	l2	Specifies the norm used in the penalization
solver	newton-cg	Specifies the algorithm to use in the optimization problem.

Table 5. Optimal hyperparameters for Random Forest.

Hyper-parameter	Optimal Value	Brief description
n_stimators	5	The number of trees in the forest
min_samples_leaf	2	The minimum number of samples required to be at a leaf node
min_samples_split	4	The minimum number of samples required to split an internal node
class_weight	balanced	Weights associated with classes. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to

		class frequencies in the input data

Table 6. Optimal hyperparameters for *k*NN.

Hyper-parameter	Optimal Value	Brief description
n_neighbors	12	Number of neighbors to classify a new sample.
weights	distance	Weight function used in prediction. With "distance", the weight between two samples is the inverse of their distance

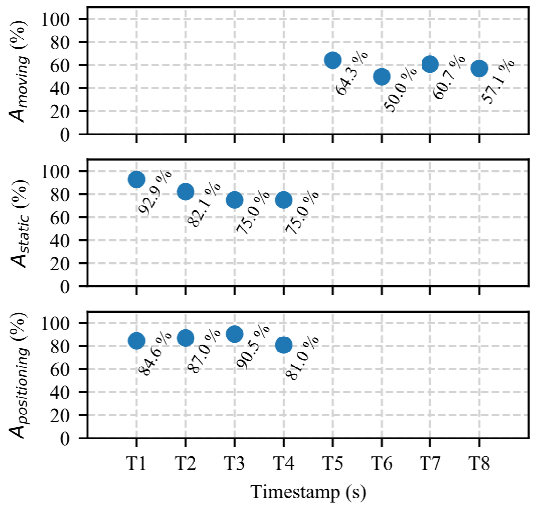


Fig. 4. Accuracies obtained using our algorithm to classify and locate pedestrians.

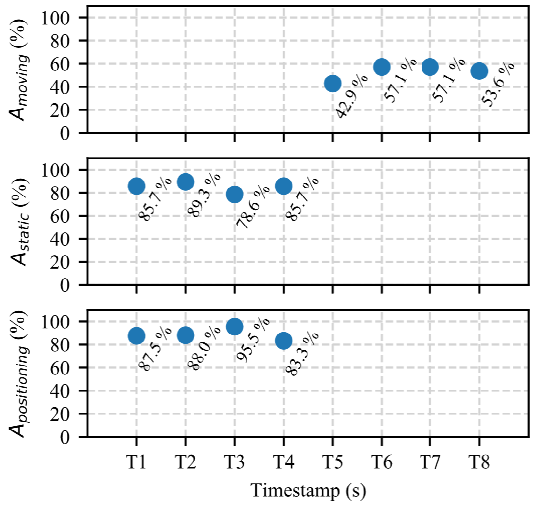


Fig. 5. Accuracies obtained using the SVC algorithm.

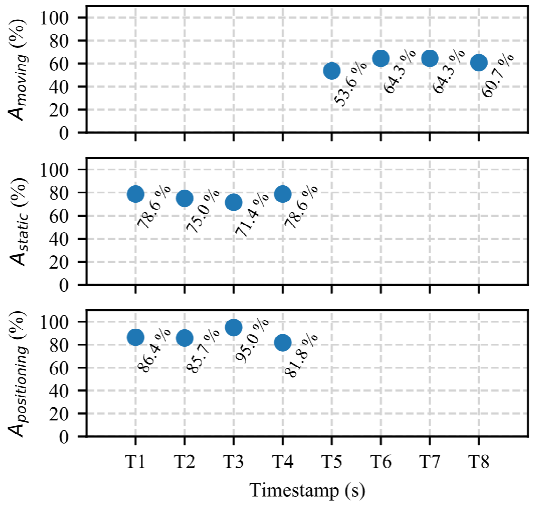


Fig. 6. Accuracies obtained using the Logistic Regression algorithm.

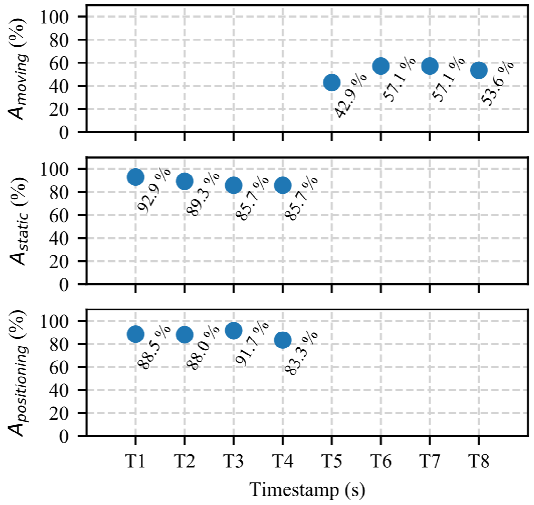


Fig. 7. Accuracies obtained by Gaussian Naive Bayes.

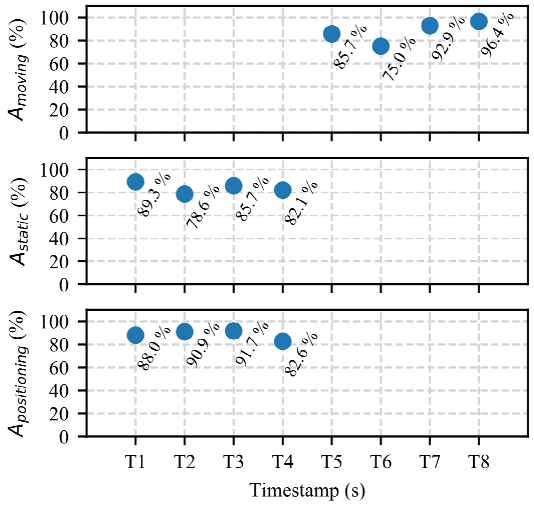


Fig. 8. Accuracies obtained by Random Forest classifier.

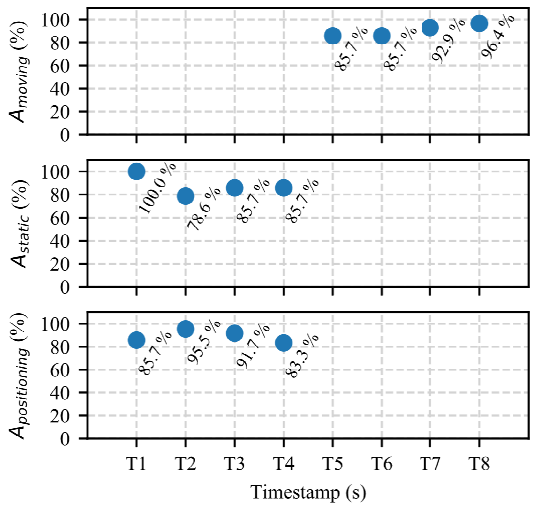


Fig. 9. Accuracies obtained by *k*NN classifier with *k*= 12.

In the light of these outcomes, we can conclude that our algorithm is able to obtain notable results in outdoor scenarios, as corroborated in the experiments. Observe that we are considering a worst-case scenario, where the amount of captured data is limited to the lowest possible time interval (as regulated by a signaled intersection). By default, the use of ML algorithms does not imply a better result. Only Random Forest and *k*NN offer a better performance. Further work is required to determine if, as expected, the computational cost of ML solutions would be a trade-off between simplicity (our proposal) and computational or energy requirements.

6 Conclusions

Simplicity, low cost, and speed of implementation are the main advantages of using WiFi-based passive methods to estimate the number of mobile devices or their behavior. Mobile phones with the WiFi activated periodically and autonomously send messages (e.g., Probe Request messages). By capturing them and processing these messages information can be extracted with different aims: distinguishing devices, tracking, estimating densities, positioning, etc. In this work we have presented the results in terms of accuracy of a WiFi-based counting people tool. Particularly, our aim was to know the behavior of pedestrians and use this information for intelligent traffic control systems in urban areas. Not only our algorithm detects pedestrians but classify them as moving or static (waiting in front of a pedestrian cross), and those that are static are also properly located. Experiments were carried out in a real outdoor scenario. The results obtained were notable. We also compared the performance of our algorithm with several machine learning algorithms. Our results were comparable to using Logistic Regression or Naive Bayes, and better than those obtained using Support Vector Classification. However, the Random Forest and *k*NN algorithms presented excellent results for this classification task. Further work is required to evaluate if the simplicity of our algorithm is enough to compensate the complexity, the computational load, or the energy requirements of using Random Forest or *k*NN.

Acknowledgments

This research was supported by the DGT Ministerio del Interior (Spain) project grant SPIP2017-02230 (STREET).

References

[1] The World Bank, “Average mobile cellular subscriptions,” [Online]. Available: https://data.worldbank.org/indicator/IT.CEL.SETS.P2. [Accessed: 01-Nov-2018]. .

[2] A. Guillen-Perez and M.-D. Cano, “A WiFi-based method to count and locate pedestrians in urban traffic scenarios,” in *WiMob 2018*, 2018, vol. 02230.

[3] R. Sanchez-Iborra and M.-D. Cano, “On the similarities between urban traffic management and communication networks: Application of the random early detection algorithm for self-regulating intersections,” *IEEE Intell. Transp. Syst. Mag.*, **9**, 4 (2017).

[4] NumPy.org, *NumPy Website*. (2018). Available at: http://www.numpy.org/ [Accessed 23 Oct. 2018].

[5] E. Jones, T. Oliphant, P. Peterson, and others, “SciPy: Open source scientific tools for Python,” *Computing in Science and Engineering*, **9**, 10–20 (2007).

[6] W. McKinney and P. D. Team, “Pandas,” *Pandas - Powerful Python Data Anal. Toolkit*, 1625, (2015).

[7] A. E. C. Redondi and M. Cesana, “Building up knowledge through passive WiFi probes,” *Comput. Commun.*, **117**, 1–12 (2018).

[8] E. Cianca, M. De Sanctis, and S. Di Domenico, “Radios as Sensors,” *IEEE Internet Things J.*, **4**, 363–373 (2017).

[9] Y. Yuan, “Crowd Monitoring Using Mobile Phones,” in *Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, 261–264 (2014)

[10] L. Schauer, M. Werner, and P. Marcus, “Estimating Crowd Densities and Pedestrian Flows Using Wi-Fi and Bluetooth,” in *MOBIQUITOUS*, 1–8 (2017).

[11] W. Pattanusorn, I. Nilkhamhang, S. Kittipiyakul, K. Ekkachai, and A. Takahashi, “Passenger estimation system using Wi-Fi probe request,” *7th Int. Conf. Inf. Commun. Technol. Embed. Syst. 2016, IC-ICTES 2016*, **2016**, 67–72 (2016).

[12] E. Vattapparamban, B. S. Çiftler, I. Güvenç, K. Akkaya, and A. Kadri, “Indoor occupancy tracking in smart buildings using passive sniffing of probe requests,” *2016 IEEE Int. Conf. Commun. Work. ICC 2016*, 38–44, (2016).

[13] L. Sun, S. Chen, Z. Zheng, and L. Xu, “Mobile

Device Passive Localization based on IEEE 802 . 11 Probe Request Frames,". **2017**, 1, (2017).

[14] A.-C. Petre, C. Chilipirea, M. Baratchi, C. Dobre, and M. Van Steen, "WiFi Tracking of Pedestrian behavior," in *Smart Sensors Networks*, Elsevier, 309–337 (2017).

[15] Y. Fukuzaki, N. Nishio, M. Mochizuki, and K. Murao, "A pedestrian flow analysis system using Wi-Fi packet sensors to a real environment," *Proc. 2014 ACM Int. Jt. Conf. Pervasive Ubiquitous Comput. Adjun. Publ. - UbiComp '14 Adjun.*, 721–730 (2014).

[16] V. Acuna, A. Kumbhar, E. Vattapparamban, F. Rajabli, and I. Güvenç, "Localization of WiFi devices using probe requests captured at unmanned aerial vehicles," *IEEE Wirel. Commun. Netw. Conf. WCNC*, (2017).

[17] A. B. M. Musa and J. Eriksson, "Tracking unmodified smartphones using wi-fi monitors," *Proc. 10th ACM Conf. Embed. Netw. Sens. Syst. - SenSys '12*, 281 (2012).

[18] A. Basalamah, "Crowd Mobility Analysis using WiFi Sniffers," *Int. J. Adv. Comput. Sci. Appl.*, **7**, 374–378 (2016).

[19] O. Kramer, "Scikit-Learn," in *Machine Learning for Evolution Strategies*, (2016).

[20] S. H. Walker and D. B. Duncan, "Estimation of the probability of an event as a function of several independent variables.," *Biometrika*, **54**, 167–179 (1967).

[21] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, **20**, 273–297 (1995).

[22] I. Rish, "An empirical study of the naive Bayes classifier," in *Workshop on empirical methods in artificial intelligence*, **22230**, pp. 41–46 (2001).

[23] T. K. Ho, "Random Decision Forests Tin Kam Ho Perceptron training," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, 278-282 (1995).

[24] B. W. Silverman and M. C. Jones, "E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951)," *Int. Stat. Rev.*, **57**, 233–238 (1989).