Pedestrian Characterization in Urban Environments Combining WiFi and AI

Antonio Guillen-Perez[1,*], Maria-Dolores Cano[1]
[1]*Department of Information and Communication Technologies, Universidad Politécnica de Cartagena, Cartagena, 30202 Spain.*
[*]*antonio.guillen@edu.upct.es*

Abstract. Knowing how many people there are in a given scenario offers new possibilities for the development of intelligent services. With this goal in mind, the use of sensors and Radio Frequency (RF) signals is becoming an interesting alternative to other classic methods such as image processing for counting people. In this paper we present a novel method for counting, characterizing, and localizing pedestrians in outdoor environments, called iPCW (intelligent Pedestrian Characterization using WiFi). iPCW is a passive, device-based sensor system that incorporates artificial intelligence techniques, more specifically, machine learning techniques. Performance evaluation using intensive computer simulations shows that iPCW achieves excellent results, with moving and static pedestrian detection accuracy above 98% and positioning accuracy above 92%.

Keywords: artificial intelligence, intelligent transportation systems, people counting, sensor systems, smart cities, WiFi;

1. Introduction

Known by different names, such as footfall counter or customer counter among others, the goal of people-counting systems is mostly the same: to obtain an estimation of the number of people in a given location with a certain degree of accuracy. Additional features such as people profiling, e.g., age, mood, behavior, etc., could also be useful depending on the final purpose desired. People-counting systems have countless applications, as summarized in Table 1: security, marketing and retail, transportation, urban planning, tourism, etc. Note that these applications are not used for people/crowd localization, although some share the same technology.

So far, people counting has mostly been based on image processing [1], on the use of sensors [2], [3] or on the use of historical and predictive models [4]. Regarding sensor methods, open environments are not always suited for sensor deployment, as dependency on specific gateways or checkpoints with an exact sensor location limits accuracy [5]. Considering image processing, most literature on people counting falls within this category. Indeed, a great deal of work has been carried out on human detection. Focusing on scientific proposals aimed at counting people and not only at detecting them, the identification of objects as human beings in an image is generally possible in two different ways: human shape detection or head/face detection. Whereas the former can only be applied in low density scenarios, the latter can be used in crowded situations. This is because, in a crowded scenario, heads are the clearest and most visible. Proposals based on human shape detection usually employ a pre-defined human template. This is normally a Gaussian or trapezoidal form representing the width, height, and size of a person. Common image processing techniques are: background subtraction to extract humans from backgrounds, quantized gradient orientation to locate interest points, regression-based techniques such as AdaBoost (Adaptive Boosting), or convolutional neural network (CNN) classifiers to indicate the type of object (e.g., a person) based on a huge dataset to allow each object to be detected [6], as well as Region-based CNN, R-CNN (Fast R-CNN [7] or Mask R-CNN [8]) to segment the image before classifying existing objects with a CNN. Despite the accuracy that these approaches can achieve, they also exhibit significant limitations. For instance, problems can arise if the people are similar to the background, and thus the background should be simple for better accuracy.

Also, it could be that the area the people occupy is too small to be recognized, or the distance between people and the camera is too large. Another drawback can arise if the people overlap is so much that those in the back and front are mistaken for one person. Finally, we can mention problems related to limited viewing angles (hidden or blind areas), changes in illumination, changes in weather conditions, privacy concerns, higher deployment costs, huge datasets, or higher computational needs of both CPU and multiple GPUs to train these algorithms, among others. On the other hand, many predictive models base their operation on a new wave of people-counting techniques; the use of radio frequency (RF) signals. In communications, the main problems with RF signals are their complex fluctuations and the multipath effect. However, what may seem to be a disadvantage in terms of effective transmission can become an asset for other services, such as crowd estimation and people counting. Recently, techniques based on the identification of RF signals and their corresponding estimation models have gained relevance. Among the advantages of these techniques is the fact that they are able to operate in longer ranges at a lower cost, and they have the ability to work through non-conducting walls and obstacles [9]. We include approaches that employ wireless sensor networks [10], [11], Bluetooth [12], [13], WiFi, cellular [14], or any other wireless technology in this category. RF-based mechanisms can use physical layer information such as Received Signal Strength Indicator (RSSI), Channel State Information (CSI), Channel Quality Indicator (CQI), or Channel Frequency Response (CFR) (see Table 2). Many of these indicators have been widely and successfully used in wireless indoor localization.

In turn, the RF-based category can be divided into two subcategories: device-free (see Figure 1) or device-based (see Figure 2). In the former, device-free approaches, the proposed techniques rely on the fact that people leave a signature, also called a fingerprint, in transmitted signals. The human body absorbs part of the signal. It becomes an obstacle and an antenna at the same time, affecting the LOS and increasing the multipath effect.

Table 1.
Applications for crowd estimation and people counting tools.

Area	Example
Energy efficiency	Adjusting the air conditioning of an indoor space (supermarkets, offices, etc.) depending on the estimated people in it, etc.
Urban planning and transportation	Pedestrian patterns, tourist flows, on-demand transport planning, etc.
Security	Crowd control (concerts, sporting events, street events, etc.), surveillance, search, and rescue, etc.
Better user satisfaction (QoS, QoE, UX)	Better planning of offered services in public areas such as airports, hospitals, amusement parks, public parks, museums, libraries, etc.
Marketing and retail	Movement patterns within a store (people tracking) or shopping center (crowd tracking), on-demand number of cashiers, etc.

Table 2.
Physical Layer Information.

Acronym	Parameter	Description
RSS	Received Signal Strength	The received signal strength in dBm of a received data frame or of a beacon measured at the receiver’s antenna.
CSI	Channel State Indicator	Channel measurements depicting the amplitudes (signal strength) and phases of every subcarrier.
CQI	Channel Quality Indicator	Current communication channel quality as measured by user equipment in cellular technologies.
CIR	Channel Impulse Response	Temporal linear filter that models the wireless propagation channel.
CFR	Channel Frequency Response	Discrete Fourier Transformation (DFT) of the Channel Impulse Response.

A complete survey of device-free activity-recognition proposals can be found in [15]. In the latter, device-based algorithms, the proposals focus on capturing/measuring information from the received signals generated by users’ devices. Despite the promising results shown in related scientific literature, it is important to note that the percentage of people who carry these signals in active mode is unknown, since users often voluntarily disconnect them in order to save energy or minimize exposure to security threads. We should therefore be aware that mobile adoption will have an impact on results in terms of accuracy (e.g., children or the elderly not carrying mobile equipment). Nevertheless, the fact is that the mobile phone is a unique device that permanently accompanies us and from which we can extract countless data on human activity [16], [17].

Under these circumstances, we present the following contributions in this paper:

1) An original passive device-based method using WiFi technology to characterize pedestrians in outdoor environments is proposed, namely iPCW (intelligent Pedestrian Characterization using WiFi). iPCW is able to differentiate between moving and static pedestrians using Machine Learning (ML) techniques, as well as the density estimation of static pedestrians, by analyzing their temporal behavior.
2) The incorporation of several ML techniques in the performance of iPCW is evaluated for comparison purposes. Particularly, we compare Logistic Regression, Gaussian Naïve Bayes, Support Vector Machine, k-Nearest Neighbor, and Random Forest. All these methods are tested with the goal of classifying pedestrians as moving or static at an intersection. The use of these ML techniques for differentiation provides excellent results and high performance, as well as great robustness against the characteristic noise present in outdoor RF environments.
3) iPCW optimization process is thoroughly described, achieving a notable performance in the defined goals of the algorithm.

The rest of the paper is organized as follows. In section 2, we review related works that deal with estimating or counting people based on WiFi technology. In section 3, we explain our proposal the ML methods tested, and the results are shown. Finally, the conclusions are reported in the last section.

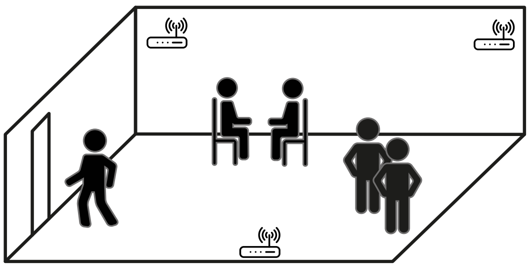


Fig 1. A general topology of the device-free category. At least one WiFi Tx and one WiFi Rx are needed; then the effect of people on the received signal is processed and associated to the number of people in the indoor area.

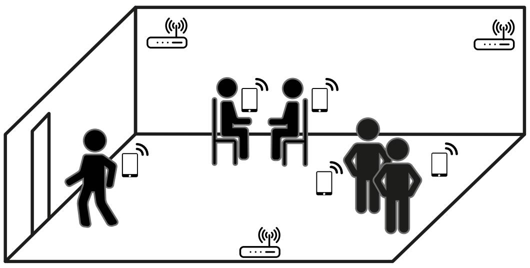


Fig 2. A general topology of the device-based category. Each person must carry at least one WiFi device from which messages will be captured and at least one WiFi Rx is needed.

2. Related Works

In this section we present a comprehensive review of the works carried out to estimate or count people based on WiFi technology.

If we focus on the proposed work, regarding the device-free approach, RSS measurements are easily available in commercial off-the-shelf WiFi cards. However, RSS variability is a well-known limitation that leads to inaccuracy in indoor scenarios [9], [15], [18]–[25]. For instance, the AP power adjustment will have an effect on RSS measurements [12]. It has also been proven that the higher the frequency the more the impact on the RSS due to human interference in the LOS [26]. As a consequence, higher frequencies are more accurate when counting people. Nevertheless, values that are too high (e.g., millimeter waves) are not expected to work as well for these applications because they often employ directional antennas and are therefore less suitable for NLOS operation.

On the other hand, channel response provides more granularity since multipath components can be differentiated [27], whereas this is not possible with RSS. In addition, we should be aware of possible

limitations regarding CSI across several frequency sub-bands with current commercial WiFi cards [28], [29]. Some advantages of using the information included in the captured beacon frames (sent by the AP) are: no need to connect to the AP, i.e., an association process is not required, no authentication is needed (it could be more useful for open environments). For applications that go beyond pure crowd estimation or people counting, bear in mind that beacons are sent by default every 100 ms in an 802.11 network. This could limit system performance in the case of movements that are too slow or too fast. The speed of most human gestures is below 10 m/s and, as explained in [30], "Assuming a space resolution of 20 cm, the human gesture should be sampled at 50 Hz. By using beacons, it is possible to sample each movement with a maximum frequency of 10 Hz (10 beacon messages per second). However, this sampling rate is enough for activity recognition/crowd counting."
In the device-based approach [5], [12], [24], [28], [29], [31], the active methods (those that force devices to send data) are less scalable and unlikely to be deployed in outdoor scenarios. Passive methods (those that simply capture messages sent by devices without forcing them) are more promising, although further work is needed, for instance, to solve the MAC randomization problem. Indeed, making these systems robust in the presence of a large number of people is still challenging.

3. iPCW: intelligent Pedestrian Characterization Using Wifi

Pedestrians should play an important role in traffic control systems, which usually focus only on vehicles and forget about the pedestrians' role. In this section, we present a passive device-based method using WiFi technology to characterize pedestrians in outdoor environments, namely, iPWC (intelligent Pedestrian Characterization using WiFi). This method aims to classifying pedestrians as moving or static, and for those that are static, provide their location. This information is expected to serve as input for an intelligent traffic management system.

3.1. Scenario description

The approach we follow to estimate the number of pedestrians involves capturing the probe request messages sent by the mobiles carried by pedestrians. These probe request messages are sent by mobile devices to discover available WiFi networks to connect to. Therefore, our proposal is within the device-based category. Because this method does not use any active mechanism to send these messages, and neither exploits any vulnerability to increase the rate of sending these messages, nor requires the pedestrian to connect to the AP, our method belongs to the WiFi-passive method. The capture of probe request messages will be carried out at short-time intervals; for a worst-case scenario; the minimum time that a traffic light is red (approximately 30 s). Therefore, we will be able to know the state of the scenario in terms of pedestrians in real time.

One of the first challenges that we must face is to differentiate pedestrians who are moving through the urban environment from other pedestrians who are static and waiting to cross at a pedestrian crossing. We will first implement a discriminator that enables us to make this classification. The behavior of the power of the probe request messages, captured from pedestrians throughout the capture interval, will be provided as the input parameter to this discriminator. Next, pedestrians classified as static will be positioned within the intersection at which they have been detected by means of the average power measured throughout the capture interval.

We model the behavior of pedestrians making the following assumptions: i) all pedestrians carry a mobile device that sends probe request messages, ii) pedestrians are either walking on the sidewalks or they are static, waiting to cross at a pedestrian crossing (pedestrian crossings are marked as ①, ②, ..., ⑧ in Figure 3), and iii) traffic lights at each intersection of the simulation scenario are synchronized and are complementary between branches; that is, if the traffic lights that control the north and south branches of an intersection i are green, the traffic lights that control the east and west branches are red. This implies that pedestrians from any pedestrian crossing will not be waiting to cross in two possible directions, since they will always have one red traffic light and one green traffic light available.

A set of simulations is carried out to obtain the widest variety of pedestrian behaviors. The simulated scenario is composed of the two intersections shown in Figure. 3. In the simulations, multiple groups of pedestrians follow different movement patterns. Basically, pedestrians are either waiting to cross (static) at any corner of any intersection or crossing an intersection at the corresponding pedestrian crossing (moving), or walking on a sidewalk. In addition to these basic cases, variations include pedestrians moving from one intersection to another, turning at the same intersection, and pedestrians moving towards a pedestrian crossing and waiting for it within the same interval of capture, this being the special case of a static pedestrian waiting to cross, who has previously moved to the intersection.

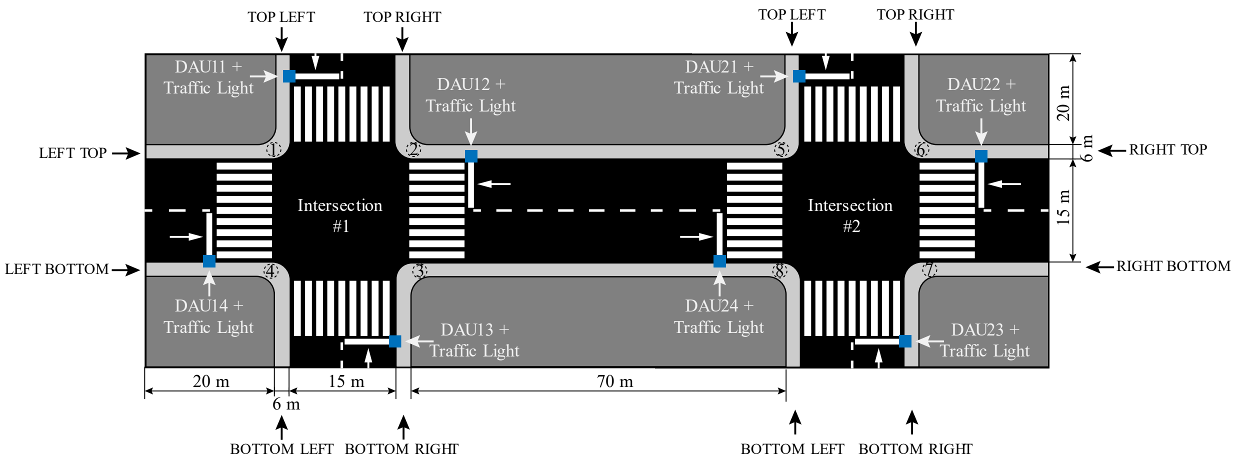


Fig.3. Scenario for the simulation analysis.

We assume there is a device in charge of the discrimination and positioning of pedestrians called Data Acquisition Unit (DAU*ij*, where *i* denotes the intersection and *j* denotes the corner of the intersection *i*). This device is placed at each corner of the intersection (4 per intersection, 8 in this simulation scenario). The DAU acts as an access point (AP), capturing probe request messages from pedestrian's mobile phones. 12 groups of pedestrians with different movement patterns are simulated, moving around all the intersections. Each group is formed of 64 pedestrians, representing a total of 768 pedestrians. We simulate 6 capture intervals of 30 seconds each, with a total simulation time of 180 s. These twelve groups of pedestrians are denoted according to their direction as:

1) From left (*l*) to right (*r*) on the upper (*u*) sidewalk, crossing the two intersections is called *Dlru*. Also, from right to left on the upper sidewalk, crossing them, is called *Drlu*.
2) From left to right on the lower (*l*) sidewalk, crossing both intersections is called *Dlrl*. Likewise, from right to left on the lower sidewalk, crossing through them, is called *Drll*.
3) From top (*t*) to bottom (*b*) along the left sidewalk at the intersection $i = 1$ is called *Dtbl1*. Similarly, from bottom to top on the left sidewalk at the intersection $i = 1$ is called *Dbtl1*.
4) From top to bottom on the right sidewalk at the intersection $i = 1$ is called *Dtbr1*. Similarly, from bottom to top on the right sidewalk at the intersection $i = 1$ is called *Dbtr1*.
5) Like the previous two, but this time, at the intersection $i = 2$, from top to bottom on the left sidewalk is called *Dtbl2*. Likewise, from bottom to top on the left sidewalk is called *Dbtl2*.
6) Finally, from top to bottom, crossing the intersection $i = 2$ on the right sidewalk is called *Dtbr2*. Likewise, from bottom to top, crossing the intersection $i = 2$ on the right sidewalk, is called *Dbtr2*.

As can be seen, by taking advantage of the symmetry of the simulation scenario, we can simplify the movements of pedestrians, being able to group them into horizontal movements and vertical movements. These two movements, despite there being slight variations within each movement, follow the patterns of movement separated by temporary marks as shown below in Table 3. All the sets of simulations are carried out using the OMNeT++ [32] simulator, with the INET framework [33]. Mobiles were modeled in OMNeT ++ using the *AdhocHost* device and DAUs with *WirelessAPWithSink*. Other important parameters of the simulator and the simulated scenario are included in Table 4. Finally, it must be noted that the proposed method has no memory, that is, we do not keep any history of the captured mobile devices and we simply classify, obtain an estimate of the number of static pedestrians, and position pedestrians in each capture interval, without taking into account the status of pedestrians captured in the previous interval.

Table 3.
Timestamps of different movements and their behavior.

Movement	Example	Time Interval	Timestamp	Behavior
Horizontal	D*lrt*	0s-30s	T1	Static in ①
		30s-60s	T2	Moving - ① to ②
		60s-90s	T3	Moving - ② to ⑤
		90s-120s	T4	Static in ⑤
		120s-150s	T5	Moving - ⑤ to ⑥
		150s-180s	T6	Static in ⑥
Vertical	D*tdl1*	0s-30s	T1	Static in ①
		30s-60s	T2	Moving - ① to ④
		60s-90s	T3	Moving - ④ to ③
		90s-120s	T4	Static in ③
		120s-150s	T5	Moving - ③ to ④
		150s-180s	T6	Static in ④

Table 4.
Simulator Parameters and their Value.

Parameters	Value
Simulation tool / Framework	OMNetT++ / INET
Version	5.2.1 / 4.0
Ground type	Flatground
Obstacle loss	Dielectric Obstacle Loss
Propagation loss	Rayleigh Fading
DAU location in height	6 m
Mobile devices location in height	1.5 m
Mobile devices motion speed	1.39 m/s
Number of mobile nodes	768
Probe Request period	2 s
Probe Request variance	0.5 s
Transmission power	13 dBm
Reception sensitivity	-120 dBm
Probability distribution to send Probe Request frames	Normal distribution with variable mean and variances

3.2. Our proposal

In this section we will describe the procedure of iPCW to, first, discriminate between moving pedestrians and static pedestrians, and then to obtain the position of the latter. The procedure is as follows:

1) After the capture interval, each DAU*ij* sends the captured messages, together with the identifier *ij* to a pre-established DAU, called DAU_*main*. There is one DAU_*main* at each intersection. After grouping all the messages, the DAU_*main* is responsible for continuing with the entire procedure
2) Pedestrians' mobile devices are differentiated by the output of applying a hash function to the MAC address included in the probe request messages. The resulting hash is called *p*.
3) For each pedestrian *p*, an array of DAUs that have detected the pedestrian *p*, is created. This array is called DAU*ijp*
4) For each DAU in the DAU*ijp* array, the temporary behavior of the probe request messages of the pedestrian *p* is obtained and their status is classified by means of the discriminator.
5) If the discriminator classifies the pedestrian *p* as moving in at least one DAU included in the DAU*ijp* array, then the pedestrian *p* is considered to be *moving*. In addition, if a DAU has only captured a single probe request message from the pedestrian *p*, this is also considered to be *moving*.
6) On the other hand, if *p* has been classified as *static* in all the DAUs included in the DAU*ijp* array, then pedestrian *p* is positioned in the DAU with the highest average power received in the entire current interval.

The pseudocode of our proposal is shown in Figure. 4. Also, the flow chart illustrating the operation of the developed method can be seen in Figure 5.

```
Algorithm 1: iPCW algorithm
    # The array_of_pedestrians detected by all DAUij is obtained.
1:  for pedestrian $p\ in\ array\_of\_pedestrians$ do:
2:      # array_of_DAUijp that have detected the pedestrian p is obtained.
3:      $static = True$ # by default, the pedestrian p is considered as static.
4:      for $DAUp\ in\ array\_of\_DAUijp$ do:
5:          # the $behavior$ is obtained
6:          $beh = classifier.evaluate(behavior)$ # Discriminator. 0 – static; 1 – moving;
7:          if $beh == 1$ then: # classifier considers p as moving.
8:              $static = False$ # pedestrian p is considered as moving.
9:          end if
10:         if $static == True$ then: # this pedestrian p is considered static in all array_of_DAUijp, then, its correct
                location corresponds to the DAUijp that presents a higher power in average.
11:             $DAUij\_max = getDAUijpMaxMeanPower(p)$ # Obtain the DAUij with max received. power
12:             $DAUij\_max.\mathrm{addPedestrian}(p)$ # Add the pedestrian.
13:         end if
14:     end for
15: end for
```

Fig. 4. Pseudocode of the proposal algorithm. iPCW.

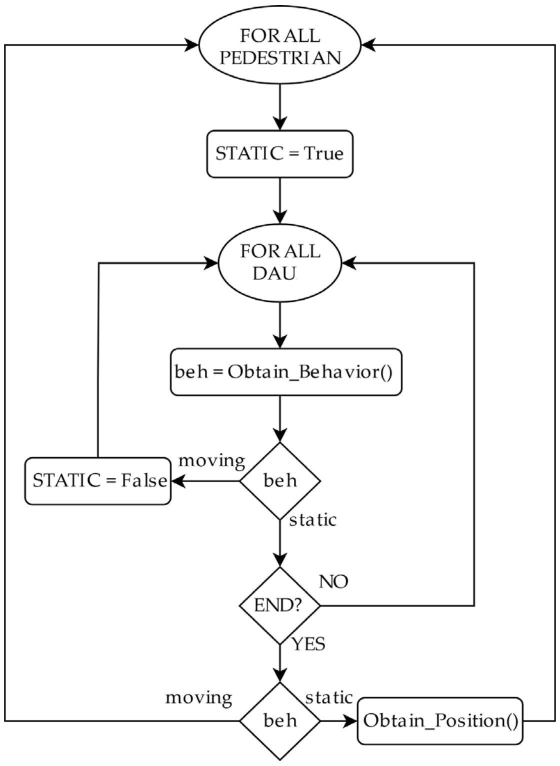


Fig. 5. Flow diagram of the proposed method (*beh*≡behavior; *DAU*≡Data Acquisition Unit).

3.3. Machine Learning Techniques for the Discriminator of iPCW

As shown in the previous section, we propose the use of a discriminator to differentiate between moving pedestrians and static pedestrians. This discriminator was implemented with ML techniques. In this section we will describe the different ML techniques tested, namely, Logistic Regressor (LR), Gaussian Naïve Bayes (GNB), Support Vector Machine (SVM), k-Nearest Neighbor (*k*NN), and Random Forest (RF), along with the procedure followed to obtain the classifier that offers the best performance in terms of classification and computation time. For a more detailed study of the ML techniques used, see [34]. That is, after training and obtaining the precision, recall, F1-score, and the execution time of the tested ML algorithms, we will select the best algorithm and will try to reduce the number of features used as input. Besides eliminating characteristics that can introduce noise, this increases precision and reduces execution times. Once the optimum features have been obtained (those that provide more information without introducing noise), a new procedure will be performed to obtain the optimal values of the hyperparameters of the selected ML algorithm. Finally, with the number of optimal features and the optimal values of the hyperparameters of the selected algorithm, we will calculate the precision, the recall, the F1-score, and the execution time in the classification of pedestrians.

3.3.1. Logistic Regressor

This algorithm is part of the family of linear regression algorithms and focuses on classification. Linear regression algorithms obtain a linear equation that most resembles training points (see Figure 6 (a)). The parameters of this linear equation are obtained by minimizing the mean square error (although it may be another parameter) between the regressed line and the training points, by means of a gradient descent algorithm (e.g., *SGD*, *adam*, *rmsprop*, etc.). To perform the classification, the logistic regressor applies a sigmoid function (also called logistic function) to the linear equation. This sigmoid activation function compresses the output data between [0,1], which, together with a decision threshold, allows for binary classification.

3.3.2. Gaussian Naïve Bayes

This is a simple, efficient, and widely used ML classification algorithm. This classifier is based on the Maximum a Posteriori (MAP) decision rule to perform classifications. It makes the "naïve" assumption that the characteristics are independent from each other, that they make the same contribution to the result, and that each characteristic can be modeled by means of a Gaussian probability distribution. These classifiers learn their internal parameters by looking at each feature of each sample separately, which allows for simple and quick training, but in certain cases where the characteristics are not independent, the accuracy obtained can be low (see Figure 6 (b)).

3.3.3. Support Vector Machine

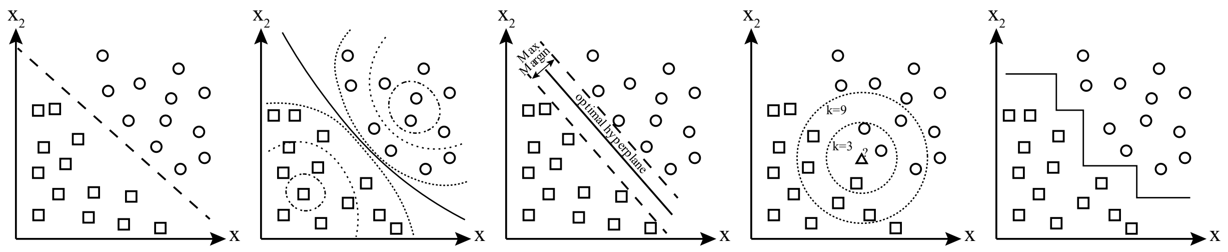
Support Vector Machine (SVM) algorithms are a set of ML algorithms widely used in classification and regression tasks, which can solve linear and non-linear problems. The operating principle of the SVM is simple: the algorithm obtains a hyperplane (in 2D it is simply a straight line), which separates the classes (see Figure 6 (c)). This hyperplane is used as a decision border and is obtained by maximizing the margin between it and the samples. The samples closest to the hyperplane are called support vectors. Thanks to the use of the trick kernel, non-linear problems can be classified, since the hyperplane used is linear by default.

3.3.4. k-Nearest Neighbor

k-Nearest Neighbor (*k*NN) is one of the simplest classification and regression algorithms in ML. *k*NN classifies new samples according to the most common class within their *k* nearest neighbors (see Figure 6 (d). Thanks to this simple operation, *k*NN is a very simple algorithm, obtaining great precision in multiple application scenarios. However, since it must keep all training samples, the algorithm is computationally expensive because it is necessary to check the distance to all the training samples for each sample to be classified.

3.3.5. Random Forest

Random Forest (RF) algorithms are an ensemble method used for both classification and regression. The ensemble methods base their operating principle on employing multiple ML methods to obtain better prediction results with broader knowledge of the data, reducing the overfitting created by the ML methods individually. In this case, random forest algorithms use multiple decision trees to create the classifier (see Figure 6 (e)). The decision trees iteratively perform two or more divisions of the training set, maximizing the accuracy of the classification as if it were done with the average value of the subgroups in each division.



(**a**) Logistic Regressor. (**b**) Gaussian Naïve Bayes. (**c**) Support Vector Machine. (**d**) k-Nearest Neighbor. (**e**) Random Forest.

Fig 6. Working principle of each of the ML algorithms studied.

3.4. Machine Learning optimization procedure

After each capture interval, the messages are grouped following the aforementioned process and a large number of statistics are obtained for each pedestrian *p* in each DAU*ijp*. These statistics are shown in Table 5. These parameters are obtained from the received power and its temporal behavior in the last capture interval. In total, there are 27 features that enable us to train the previously mentioned ML classification algorithms.

To measure the performance of the different ML algorithms, we will use a confusion matrix. In the cells of this matrix, the number of predictions (and the percentage) are shown when classifying data of different classes (if it is a binary classification: negative and positive), depending on whether the classification is correct (true classification) or incorrect (false classification). Each column of this matrix represents the predictions of the algorithm for each class, while each row represents the actual class. An example of a binary confusion matrix can be seen in Table 6. From a confusion matrix we can see that:

- TN (True Negative): is an outcome where the model correctly predicts the negative class.
- TP (True Positive): is an outcome where the model correctly predicts the positive class.
- FN (False Negative): is an outcome where the model incorrectly predicts the negative class.
- FP (False Positive): is an outcome where the model incorrectly predicts the positive class.

From the results obtained in this matrix and the various definitions of the classifications shown, we can obtain various metrics offering more in-depth knowledge of the performance of the algorithms in classification tasks. The first parameter we can obtain is precision. Precision is defined as the ratio between True Positives (TP) and the sum of these and False Positives (FP). The formula for precision can be seen in (1). This parameter is very useful for tasks where it is important to control FP. For instance, in email spam detection, an FP means that an email that is not spam (negative class) has been classified as spam (predicted as positive) and an important email might not be read by the user.

On the other hand, we have recall. This parameter indicates the ratio between True Positives (TP) and the sum of these and False Negatives (FN). This parameter is very important, for example in tasks of fraud detection or the diagnosis of diseases. To classify a positive case (of fraud or disease) a negative should be penalized and therefore, we will try to maximize the recall. The recall formula is indicated in (2). Since there is a compromise between both values (see precision-recall tradeoff [35]), a last parameter is defined, which is called F1-score. This parameter is the geometric mean of precision and recall. Using this parameter, it is easier to obtain a general idea of the performance of a classification algorithm, since the F1-score is useful when a balance between precision and recall is necessary. The formula that follows this parameter is shown in equation (3). The metric that indicates the number of occurrences of each class to predict is called support. This metric is useful to get an idea of whether it is a problem where there is a similar number of occurrences on each class or not.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

Table 5.
Statistics used.

Statistic	
	Mean power
	Variance
From linear regression line	Slope
	Intercept
	r-value
From polynomial regression of degree 2	Coefficient degree 0
	Coefficient degree 1
	Coefficient degree 2
	Residuals
From polynomial regression of degree 3	Coefficient degree 0
	Coefficient degree 1
	Coefficient degree 2
	Coefficient degree 3
	Residuals
From polynomial regression of degree 4	Coefficient degree 0
	Coefficient degree 1
	Coefficient degree 2
	Coefficient degree 3
	Coefficient degree 4
	Residuals
	Kurtosis parameter
Quantile	25%
	50%
	75%
	Pearson correlation coefficient
	Pearson p-value coefficient
	Skewness parameter

Table 6.
Example of Binary Confusion Matrix.

	Predicted: negative (0)	Predicted: positive (1)
Actual: negative (0)	TN	FP
Actual: positive (1)	FN	TP

TN = True Negative, FP = False Negative, FN = False Negative, TP = True Positive.

3.5. Machine Learning optimization results

In this section we will see and discuss in detail the results obtained in the discrimination process for iPCW. The dataset used was made up of the aforementioned simulations, and it consisted of 36,864 individual samples, each with 27 characteristics. These 36,864 samples come from the behavior of 768 pedestrians (64 pedestrians / MAC addresses x 12 directions) in 6 capture intervals, in the 8 DAU*ij* (768x6x8 = 36,864). Half of it (18,432 samples) corresponds to static pedestrian behavior and the other half to moving pedestrians. Of the complete dataset, 65% (23,961) was used for the training set and the remaining 35% (12,903) for the test set. For the training set there were 12,722 occurrences for the static class and 11,139 for the moving class, and for the test set there were 6,508 occurrences for the static class and 6,395 for the moving class.

The first task was to compare the different aforementioned ML algorithms with all the input features. In order to obtain the optimal values of the hyperparameters of the algorithms that maximize the F1-score parameter, a cross-validation grid search procedure was performed (*gridsearchcv*). The optimal values of the hyperparameters are shown in Table 7. The confusion matrix is shown in Table 8, and Table 9 shows the precision, recall, F1-score, and support. Finally, the execution time of the algorithms is shown in Table 10, where the mean time of 100 executions of the classification algorithm and their standard deviations, both expressed in milliseconds, are indicated.

In view of the results, it can be concluded that the algorithm that achieves the best performance is the Random Forest classifier, with a reasonable execution time (less than 500 ms) and precision, recall, and F1-score higher than 99.6%

In order to reduce the execution time to a minimum and obtain a light algorithm that can be executed in embedded systems, we followed an additional procedure that involved selecting the characteristics that provide more information. To do so, a Recursive Feature Elimination mechanism with Cross-Validation (RFECV) was used. This recursive procedure ranks the characteristics according to the information they provide individually, and then eliminates the feature that provides the least information. After this elimination, classification is performed. This procedure is carried out until there is a single characteristic or a predefined minimum number. Thanks to this, the variables that are not contributing information or that are noise can be eliminated. The different classification metrics and reducing the execution time are also improved.

Table 7.
Optimal Hyperparameters of all ML algorithms tested.

	Hyper-parameter	Optimal value	Brief description
LR	C	4.25e-4	Inverse of regularization strength.
	penalty	l2	Specifies the norm used in the penalization.
	Solver	newton-cg	Specifies the algorithm to use in the optimization problem.
SVM	C	0.85	Penalty parameter C of the error term.
	gamma	0.01	Kernel coefficient.
	kernel	rbf	Specifies the kernel type to be used in the algorithm.
*k*NN	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.
RF	n_estimators	26	The number of decision trees in the random 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.

LR = Linear Regressor, SVM = Support Vector Machine, *k*NN = *k*-Nearest Neighbor, RF = Random Forest.

Table 8.
Confusion Matrix of all ML algorithms tested.

		Predicted: Static	Predicted: Moving
LR	True: Static	4294 / 33.28%	2214 / 17.15%
	True: Moving	2083 / 16.14%	4312 / 33.42%
GNB	True: Static	3918 / 30.37%	2590 / 20.07%
	True: Moving	1903 / 14.75%	4492 / 34.81%
SVM	True: Static	4907 / 38.03%	1601 / 12.41%
	True: Moving	2042 / 15.83%	4353 / 33.74%
*k*NN	True: Static	6353 / 49.24%	155 / 1.20%
	True: Moving	448 / 3.47%	5947 / 46.09%
RF	True: Static	**6490 / 50.31%**	**18 / 0.14%**
	True: Moving	**20 / 0.15%**	**6373 / 49.39%**

LR = Linear Regressor, GNB = Gaussian Naïve Bayes, SVM = Support Vector Machine, *k*NN = *k*-Nearest Neighbor, RF = Random Forest.

Table 9.
Classification report of all ML algorithms tested.

	State	Precision	Recall	F1-score
LR	Static	0.6687	0.6599	0.6643
	Moving	0.6655	0.6743	0.6699
	avg/total	0.6671	0.6671	0.6671
GNB	Static	0.6684	0.6120	0.6390
	Moving	0.6392	0.7024	0.6693
	avg/total	0.6538	0.6472	0.6542
SVM	Static	0.7017	0.7541	0.7299
	Moving	0.7353	0.6806	0.7069
	avg/total	0.7185	0.7174	0.7184
*k*NN	Static	0.9329	0.9762	0.9541
	Moving	0.9752	0.9300	0.9521
	avg/total	0.9540	0.9531	0.9535
RF	Static	0.9965	0.9972	0.9968
	Moving	0.9973	0.9965	0.9969
	avg/total	0.9969	0.9969	0.9969

LR = Linear Regressor, GNB = Gaussian Naïve Bayes, SVM = Support Vector Machine, *k*NN = *k*-Nearest Neighbor, RF = Random Forest.

Table 10.
Execution Time of all ML algorithms tested.

	Execution time (mean ± std) (100 runs)
LR	7.2389 ms ± 0.2451 ms per run
GNB	108.0081 ms ± 0.1914 ms per run
SVM	92970.1549 ms ± 5779.2150 ms per run
*k*NN	35244.3757 ms ± 2799.9939 ms per run
RF	**311.8065 ms ± 10.1476 ms** per run

LR = Linear Regressor, GNB = Gaussian Naïve Bayes, SVM = Support Vector Machine, *k*NN = *k*-Nearest Neighbor, RF = Random Forest.

The results showed that the best results were obtained with the following 5 characteristics shown in Table 11. This can be seen in Figure 7, where the CV-score accuracy is indicated according to the number of characteristics that are considered. After this procedure of selecting the characteristics that provide the most information, the operation was again carried out to obtain the optimal values of the hyperparameters, but this time only for the previously selected Random Forest algorithm. The optimal values of the hyperparameters are shown in Table 12.

The confusion matrix is shown in Table 13. A summary report shows the main performance metrics in Table 14. Finally, the average computation time and its standard deviation of 100 executions, both expressed in milliseconds, are shown in Table 15.

Table 11.
Statistics used after RFECV.

Statistic	
From polynomial regression of degree 3	Coefficient degree 0
	Coefficient degree 3
From polynomial regression of degree 4	Coefficient degree 0
	Coefficient degree 3
	Coefficient degree 4

Table 12.

Optimal Hyperparameters of Random Forest after RFECV.

	Hyper-parameter	Optimal Value	Brief description
RF	n_estimators	5	The number of decision trees in the random forest.
	min samples leaf	1	The minimum number of samples required to be at a leaf node.
	min samples split	2	The minimum number of samples required to split an internal node.

RF = Random Forest.

Table 13.

Confusion Matrix of all ML algorithms tested.

		Predicted: Static	Predicted: Moving
RF	True: Static	6495 / 50.34%	13 / 0.10%
	True: Moving	6 / 0.05%	6389 / 49.52%

RF = Random Forest.

Table 14.

Classification report of all ML algorithms tested.

	State	Precision	Recall	F1-score	Support
RF	Static	0.9991	0.9980	0.9985	6508
	Moving	0.9980	0.9991	0.9985	6395
	avg/total	0.9986	0.9986	0.9985	12903

RF = Random Forest.

Table 15.

Execution Time of all ML algorithms tested.

	Execution time (mean ± std):
RF	110.7994 ms ± 0.6019 ms per run (100 runs)

RF = Random Forest.

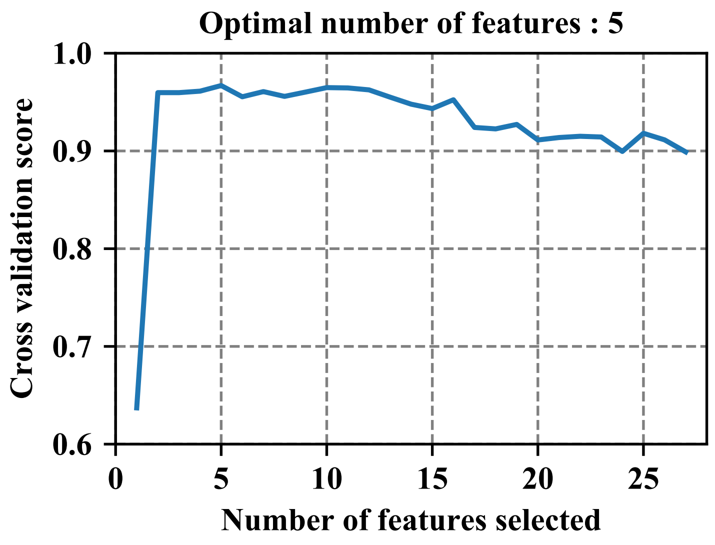


Figure 7. RFECV results. The optimal number of features are 5. By increasing the number of characteristics introduced in the algorithm, the cross-validation score is reduced. This is because not all the characteristics provide the same information, in addition to the possibility that they are contradicting themselves or can add noise in the classification.

The benefit of selecting the characteristics that provide more information (using the RFECV procedure) can clearly be seen in the results obtained (see Figure 8), since the metrics used have increased their value, exceeding 99.8% in precision, recall and F1-score, and the execution time has been reduced to one third of its original value.

3.6. Performance of iPCW

Having seen the algorithm responsible for discrimination between moving pedestrians and static pedestrians, we now show how to locate those pedestrians that were classified as static. The static pedestrians will simply be placed in the DAU*ijp* that has received more average power in the capture interval. The performance of iPCW will be evaluated in terms of discrimination (classification) and positioning accuracy, as described in (4) and (5), where:

- TN (True Negative): is an outcome where the classifier *correctly* predicts the *static* state.
- TP (True Positive): is an outcome where the classifier *correctly* predicts the *moving* state.
- FN (False Negative): is an outcome where the classifier *incorrectly* predicts the *static* state.
- FP (False Positive): is an outcome where the classifier *incorrectly* predicts the *moving* state.
- *pedestrians_located_properly*: is the number of pedestrians correctly located at the crossing they are waiting to cross.
- *pedestrians_detected_as_static*: is the number of pedestrians detected as static by the discriminator.

iPCW is tested in the same scenario previously explained (Figure 3). The results of both the classification of pedestrians in the static and moving classes, as well as the positioning, are shown in Figure 8. The mechanism proposed for the classification of pedestrians with different behavior in static and moving classes, as well as for positioning static pedestrians at intersections, obtains a high degree of precision. This accuracy is above 98% in the classification task and above 92% in the positioning task. The positioning accuracy only appears in the time intervals where there are static pedestrians (T1, T5 and T6). As can be seen in Figure 8, the classification accuracy between static and moving pedestrians is over 98% in all time intervals. On the other hand, if we look at the positioning accuracy, for the time intervals where the pedestrians were static, and therefore can be positioned, this accuracy is higher than 92% in all time intervals (T1, T5, and T6).

$$Aclassification = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Apositioning = \frac{pedestrians\_located\_properly}{pedestrians\_detected\_as\_static} \tag{5}$$

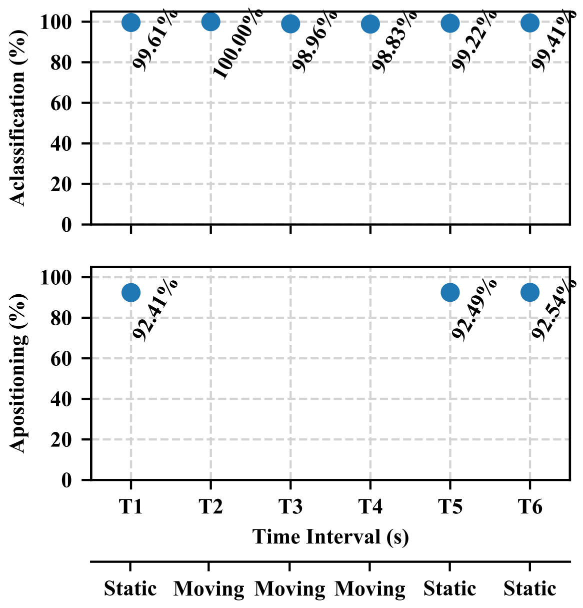


Figure 8. Results of classification accuracy and positioning accuracy. The positioning accuracy only appears in the time intervals where there are static pedestrians (T1, T5, and T6).

As a final note, we have identified some drawbacks and possible countermeasures to using iPCW. Regarding MAC randomization, we should measure the likelihood of randomizing a MAC address within the measurement interval, i.e., what the likelihood is that a WiFi device changes the MAC during the measurement interval. In our case study, measurement intervals are quite short (corresponding to the red duration of a traffic light), ranging approximately between 40 s and 100 s, reducing the likelihood of this occurring. Another problem could be a mobile device with a low frequency of Probe Request messages. To address this point and minimize its impact, we plan to calculate the minimum frequency of sending Probe requests, so that our system works with an accuracy greater than a given value. This frequency will be a function of the measurement period. Finally, it is correct that iPCW does not count people but WiFi devices. However, this is enough to obtain an estimation of people density that, in turn, is highly useful to serve as input for an intelligent traffic light system.

4. Conclusions

In this paper, we have proposed an original technique to count, characterize, and locate pedestrians in urban environments. After reviewing previous works to clearly understand the state of the art, the new method has been introduced; specifically, a passive device-based method using WiFi technology called iPCW. After comparing the performance of iPCW using different ML algorithms, the best results have been achieved using Random Forest for iPCW. All the performance evaluations have been carried out with intensive computer simulations in an urban scenario. The results have shown that our proposal obtain an excellent performance, with a detection accuracy between moving pedestrians and static pedestrians above 98% and with a positioning accuracy higher than 92%. As future work, we plan to implement iPCW in real devices and experimentally test its performance.

Acknowledgments: This work was supported by DGT Ministerio del Interior (Spain) under Project SPIP2017-02230; by the AEI/FEDER EU project grant TEC2016-76465-C2-1-R (AIM), and by a pre-doctoral grant 20740/FPI/18. Fundación Séneca. Región de Murcia (Spain).

Conflicts of Interest: The authors declare no conflict of interest.

Biographical notes: Antonio Guillen-Perez received the B.Sc. degree in telecommunication engineering in 2016 and the M.Sc. degree in information and communication technologies in 2018 from the Universidad Politécnica de Cartagena, Murcia, Spain, where he is currently working toward the Ph.D. Degree with the Department of Information and Communication Technologies. His main research interests are Intelligent Transport Systems (ITS), artificial intelligence (AI), deep learning (DL), multi-agent systems, multiagent deep reinforcement learning, autonomous cars, IoT and smart cities. He has published several papers in national and international conferences and journals. He has collaborated as a reviewer for international conferences and is a member of many IEEE technical committees.

Maria-Dolores Cano Cano received the telecommunications engineering degree in 2000 from Universidad Politécnica de Valencia, Spain, and the Ph.D. from Universidad Politécnica de Cartagena (UPCT), Spain, in 2004. She joined UPCT in 2000, where she is an Associate Professor in the Department of Technologies and Communications. She has published numerous research works in international journals and conferences in the areas of Quality of Service (QoS)/Quality of user Experience (QoE), Intelligent Transportation Systems, IoT, and security provisioning. Dr. Cano was awarded a Fulbright grant as a Postdoctoral Researcher in 2006 at Columbia University, New York, NY, USA, and has also collaborated since 2007 with several universities in South America. She received the Best Paper Award at the 10th IEEE International Symposium on Computers and Communications in 2005 and the Exemplary Reviewer Recognition by IEEE Communication Letters in 2010, among other recognitions.

References

[1] J. Garcia, A. Gardel, I. Bravo, J. L. Lázaro, M. Martínez, and D. Rodríguez, "Directional People Counter Based on Head Tracking," *IEEE Trans. Ind. Electron.*, vol. 60, no. 9, pp. 3991–4000, 2013.

[2] M. Stubenschrott, T. Matyus, and C. Kogler, "Real-Time Estimation of Pedestrian Inflow Rates from Saturated Sensor Counting Data in a Complex Metro Station," in *Proc. IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 1958–1963.

[3] N. Ahmed, A. Ghose, A. K. Agrawal, C. Bhaumik, V. Chandel, and A. Kumar, "SmartEvacTrak: A People Counting and Coarse-Level Localization Solution for Efficient Evacuation of Large Buildings," in *12nd nternational Workshop on Crowd Assisted Sensing Pervasive Systems and Communications*, 2015, pp. 372–377.

[4] Z. Duan, L. Liu, and S. Wang, "MobilePulse: Dynamic profiling of land use pattern and OD matrix estimation from 10

million individual cell phone records in Shanghai," in *Proc. International Conference on Geoinformatics*, 2011, pp. 1–6.

[5] H. Li, E. C.L. Chan, X. Guo, J. Xiao, K. Wu, and L. M. Ni, "Wi-Counter: Smartphone-Based People Counter Using Crowdsourced Wi-Fi Signal Data," *Trans. Hum. Mach. Syst.*, vol. 45, no. 4, pp. 442–452, 2015.

[6] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *CoRR*, vol. abs/1804.0, 2018.

[7] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[8] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[9] C. Xu *et al.*, "SCPL: Indoor device-free multi-subject count- ing and localization using radio signal strength," in *Proc. ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2013, pp. 79–90.

[10] S. H. Doon, "Spectral Human Flow Counting with RSSI in Wireless Sensor Networks," in *International Conference on Distirbuted Computing in Sensor Systems*, 2016, pp. 110–112.

[11] S. Y. Fadhlullah and W. Ismail, "A Statistical Approach in Designing an RF-Based Human Crowd Density Estimation System," *Int. J. Distrib. Sens. Networks*, vol. 2016, no. Article ID 8351017, pp. 1–9, 2016.

[12] Y. Yuan, "Crowd Monitoring Using Mobile Phones," in *Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, 2014, pp. 261–264.

[13] L. Schauer, M. Werner, and P. Marcus, "Estimating Crowd Densities and Pedestrian Flows Using Wi-Fi and Bluetooth," in *MOBIQUITOUS*, 2014, pp. 1–8.

[14] C.-Y. Chiang, J.-Y. Chuang, J.-K. Chen, C.-C. Hung, W.-H. Chen, and K.-R. Lo, "Estimating Instant Traffic Information by Identifying Handover Patterns of UMTS Signals," in *Proc. 14th International IEEE Conference on Intelligent Transportation Systems*, 2011, pp. 390–395.

[15] E. Cianca, M. De Sanctis, and S. Di Domenico, "Radios as Sensors," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 363–373, 2017.

[16] K. Farrahi and D. Gatica-Perez, "Probabilistic Mining of Socio-Geographic Routines From Mobile Phone Data," *IEEE J. Sel. Top. Signal Process.*, vol. 4, no. 4, pp. 746–755, 2010.

[17] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7657 LNCS, pp. 216–223.

[18] M.Nakatsuka, H.Iwatani, and J.Katto, "A study on passive crowd density estimation using wireless sensors," in *Proc. Int. Conf. Mobile Comput. Ubiquitous Netw.*, 2008, pp. 1–7.

[19] T. Yoshida and Y. Taniguchi, "Estimating the number of people using existing WiFi access point in indoor environment," in *6th European Conference of Computer Science ECCS '15*, 2015, pp. 46–53.

[20] S. Depatla, A. Muralidharan, and Y. Mostofi, "Occupancy Estimation Using Only WiFi Power Measurements," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 7, pp. 1381–1393, 2015.

[21] W. Xi *et al.*, "Electronic Frog Eye: Counting Crowd Using WiFi," in *IEEE Conference on Computer Communicaitons INFOCOM*, 2014.

[22] S. Di Domenico, G. Pecoraro, E. Cianca, and M. De Sanctis, "Trained-once device-free crowd counting and occupancy estimation using WiFi: A Doppler spectrum based approach," *Int. Conf. Wirel. Mob. Comput. Netw. Commun.*, 2016.

[23] M. T. Hoang, B. Yuen, X. Dong, T. Lu, R. Westendorp, and K. Reddy, "Recurrent Neural Networks for Accurate RSSI Indoor Localization," *IEEE Internet Things J.*, 2019.

[24] C. H. Hsieh, J. Y. Chen, and B. H. Nien, "Deep Learning-Based Indoor Localization Using Received Signal Strength and Channel State Information," *IEEE Access*, 2019.

[25] A. Zanella, "Best Practice in RSS Measurements and Ranging," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 4, pp. 2662–2686, 2016.

[26] K.Woyach, M.Puccinelli, and D.Haenggi, "Sensorless sensing in wireless networks: Implementation and measurements," in *Proc. 2nd Int. Workshop Wireless Netw. Measur. (WiNMee)*, 2006, pp. 77–89.

[27] Z. Yang, Z. Zhou, and Y. Liu, "From RSSI to CSI: Indoor Localization via Channel Response," *ACM Comput. Surv.*, vol. 46, no. 2, pp. 25–32, 2013.

[28] H. Chen, Y. Zhang, W. Li, X. Tao, and P. Zhang, "ConFi: Convolutional Neural Networks Based Indoor Wi-Fi Localization Using Channel State Information," *IEEE Access*, 2017.

[29] J. P. Conti, T. B. N. da Silveira, and D. P. Araújo, "Dynamic crowd counting via 802.11 MAC layer," in *IEEE International Symposium on Consumer Electronics*, 2016, pp. 1–2.

[30] M. De Sanctis, E. Cianca, S. Di Domenico, D. Provenziani, G. Bianchi, and M. Ruggieri, "WIBECAM: Device free human activity recognition through WiFi beacon-enabled camera," in *Proc. 2nd Workshop Phys. Anal. (WPA)*, 2015, pp. 7–12.

[31] R. Buchakchiev, "People density estimation using Wi-Fi infrastructure," Aalborg University, 2016.

[32] "OMNeT++ Discrete Event Simulator - Home Page." [Online]. Available: https://www.omnetpp.org/. [Accessed: 17-Jul-2018].

[33] "INET Framework." [Online]. Available: https://inet.omnetpp.org/. [Accessed: 28-Aug-2019].

[34] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *Eurasip Journal on Advances in Signal Processing*. 2016.

[35] M. Buckland and F. Gey, "The relationship between Recall and Precision," *J. Am. Soc. Inf. Sci.*, vol. 45, no. 1, pp. 12–19, 1994.