Abstract—The indoor location market is forecasted by ABI Research to reach approximately $10B by 2020. However, this emerging area, suffers from limited positioning accuracy induced by the many existing indoor sensing platforms. We present novel noise-reduction and data-smoothing algorithms, designed to cleanse Wi-Fi-based indoor positioning data. An empirical evaluation of these algorithms demonstrated up to 75.5% improvement in the data accuracy. This improved accuracy opens new market opportunities in the retail and travel and transportation domains.

I. INTRODUCTION

An Indoor Positioning Systems (IPS)[1] can locate mobile devices inside a building using radio waves, magnetic fields, acoustic signals, or other sensory information collected by mobile devices. In recent years, radio waves-based IPS systems have become very popular as they can locate smartphones by tracking the device’s Wi-Fi or BLE (Bluetooth Low Energy) signals. Together with mobile, this position tracking can be leveraged to support high-value services such as indoor location-based analytics (e.g., understanding customer traffic), real-time engagements (e.g., sending the customer a mobile coupon for a shirt when the customer is standing near it), asset management (e.g., tracking medical equipment in hospitals [2]) and indoor navigation (e.g., in hospitals and malls). These services and solutions can transform the user experience for customers in the retail and travel and transportation domains. Enterprises can also leverage indoor location data to improve business insights and build new engagement models with customers. In the recent report by ABI Research, the market for new indoor location-based business opportunities is expected to reach approximately $10B by 2020 [3].

Compared to outdoor positioning, indoor positioning is inherently more complex and less accurate for two main reasons. First, in an indoor venue, the signals are attenuated and scattered by static obstacles such as walls and “dynamic obstacles” such as people. This is different from outdoors, where the lines of sight from the GPS satellites are usually visible. Second, in an indoor venue there is typically no fixed network of roads, unlike the outdoor roads network that can be leveraged (e.g., by navigation applications applying snap-to-grid methods) to achieve high positioning accuracy. These challenges motivate the use of data cleansing algorithms to improve the accuracy of indoor location data.

The most popular indoor localization systems use Wi-Fi-based technologies since they i) can be based on existing Wi-Fi networks, which are widely available [4] [5]; and ii) do not require the device or person to take action such as opting-into the sensing or installing a mobile application. While there are numerous approaches for Wi-Fi-based positioning, virtually all of them are not applicable for wide commercial use since they are either i) not sufficiently accurate (e.g., the RSSI-based approach); or ii) complex and costly to maintain (e.g., the fingerprints-based and Time of Flight-based approaches); or iii) incur prohibitive costs by requiring a big installation (e.g., the Angle of Arrival-based approach [6]).

Over the past two years, beacons-based indoor positioning systems have emerged as a cheap and easy-to-maintain alternative to the Wi-Fi-based ones [7], [5]. These systems, however, require that a mobile application is installed on the customer device to collect the beacon’s (sensors) signals and infer the device’s positioning; Alternatively the application can send the signals to a backend system for such an inference. This requirement is a major limitation. According to a recent survey [8], only 5% of the customers are willing to install such a mobile application. Conversely, 65% of the people have their smartphones Wi-Fi module enabled at all times, which is sufficient for Wi-Fi-based positioning.

Trajectory analytics has received a lot of attention in recent years. In particular, smoothing trajectory data has been a primary focus of applications in this domain. One of the main motivations is the need to clean the noise, which is a result of imprecise data gathering [9]. Other motivations include simplification of the data for computation and visualization purposes [10], feature extraction [11], snapping to layout representation (e.g.,
in road networks) [12] and more. However, most of the work in this area has been done on either outdoor or abstract data, where the first is collected with satellite navigation systems and the second is not specified to the environment but rather defined abstractly. We are not aware of any work devoted to smoothing indoor trajectories, taking into account the special characteristics of this kind of data (see Section II).

In the indoor domain, a lot of work has been devoted to improving localization. Lymberopoulos et al. [13] offer an excellent review of the latest technologies used in this domain as well as the related challenges. Using data that includes the identification of the mobile entities is necessary to generate trajectories. As we show in this paper, analyzing the locations as part of a trajectory is imperative. No other work in this area addresses this challenge. Although, trajectories have been of interest in the indoor domain in several publications (e.g., [14]), most of this work has been devoted to extracting trajectories for navigation purposes. This is a completely different task from the one we investigate.

In this paper, we present Weighted Positioning (WP), a real-time data cleansing algorithm designed to tackle Wi-Fi-based positioning inaccuracies. This algorithm is composed from two sub-algorithms: noise reduction and data smoothing. The noise reduction algorithm detects and removes noisy samples from the positioning samples set. The data smoothing algorithm adjusts and recalculates the remaining positioning samples to determine average positioning, based on each sample’s timestamp and RSSI weights, within a given time interval (called the data smoothing window). The data smoothing algorithm works to overcome inherent positioning inaccuracies that are too small to be considered noise yet can still be a few meters from the device’s actual positioning.

This paper proceeds as follows: Section II describes the WP algorithm and the theory on which it is based. Section III describes the empirical evaluation of this algorithm in both static and dynamic settings. These, demonstrate an improvement in the data accuracy of up to 75.5%. Finally, Section IV concludes the paper.

II. THE WP ALGORITHM: NOISE REDUCTION AND DATA SMOOTHING

A. Noise Reduction

A Wi-Fi-based positioning system has an inherent positioning inaccuracy due to the technology limitations described in the previous section. Hence, Wi-Fi-based positioning vendors define an inaccuracy radius \( R \) of about 3 to 5 meters. They claim that, for a given device, at least 75% of the positioning samples will be within an inaccuracy disk/circle whose center is the device’s coordinates \((X_i, Y_i)\) and its radius is \( R \). We then denote the disk by \( D = ((X_i, Y_i), R) \). A positioning sample that is within \( D \) is considered as a valid and one outside \( D \) is considered a noisy sample. We note that in a commercial deployment such as a shopping mall, for dynamic devices like smartphones, the sensing system cannot infer whether a sample is noisy, since the exact positioning of the device is unknown. While the exact classification of noisy positioning samples is not feasible for customer device, we describe a model that can support highly accurate classification based on a moving inaccuracy disk.

To explain this model, we consider the following example: assume that at time \( t_i \) a mobile device \( DEVICE_i \) is at coordinates \((X_i, Y_i)\). A positioning sample \( P_i \) received at this time can be anywhere within \( D = ((X_i, Y_i), R) \) and still considered a valid (i.e., non-noisy) sample. Assume that the subsequent positioning sample \( P_j \) with coordinates \((X_j, Y_j)\) is received at time \( t_j > t_i \). Assume also that i) \( P_i \) is a valid sample (we soon relax this assumption); and ii) each mobile device can move at a speed lower than or equal to \( S \) kph (kilometer per hour). According to the latter assumption, the Euclidean distance between \( P_i \) and \( P_j \) denoted as \( Distance_{i,j} \) divided by the time gap between these two samples \((t_j - t_i)\) should be \( \leq S \). We note, however, that the reported \( P_i \) and \( P_j \) coordinates can drift by at most \( R \) meters in both the \( X \) and \( Y \) directions yet still be considered valid samples. Hence, we need to relax the above speed condition to include a drift of up to \( R \) meters from each sample, i.e., \( \frac{Distance_{i,j}}{t_j - t_i} \leq S + 2R \). If this condition is not met, \( P_j \) is classified as a noisy sample. Finally, we relax the assumption that \( P_i \) is a valid sample by replacing it with the device’s estimated positioning (WP described in the next section); this is the average of all the positioning samples within the data smoothing window. This estimated positioning is maintained by the data smoothing algorithm described in the next section.

For Parameter settings, we typically set \( R \) to be “a little bit less” than the claimed inaccuracy radius defined by the sensor vendor, e.g., 2.5 meters. We also typically set \( S \) (which is the upper bound on the speed at which a mobile device is “allowed” to move) as the average walking speed of a person, which is around 4.5 kph. We use these conservative values for the following reasons: First, in the above speed condition we are subtracting \( 2R \) from the “allowed” distance between \( P_i \) and \( P_j \). This subtraction implies the worst case (and not
common) scenario in which both \( P_1 \) and \( P_2 \) drift (from a positioning perspective) by \( R \) meters to exactly opposing directions, whereas typically the combined drift will be less than \( 2R \). Second, mobile devices are carried by people, who usually do not walk in straight lines. Moreover, in an indoor venue most of the devices/people are static or “semi-static” for long periods. These two arguments imply an effective smaller inaccuracy radius and allowed speed.

In Section III, we present our empirical evaluation of the noise reduction algorithm with different \( R \) and \( S \) values. This evaluation demonstrates that the aforementioned settings achieve high classification accuracy. On one hand the algorithm detects the vast majority of the noisy samples and on the other, it does not classify valid samples as noisy ones.

### B. Smoothing

For each device \( D_i \), the data smoothing algorithm maintains its estimated positioning, denoted as weighted positioning. This positioning is an average of valid positioning samples of \( D_i \) received within a time window. The weight of each such sample is composed from a time weight and an RSSI weight. There can be different strategies for calculating each such weight, though there should be two main rules of thumb: a later sample should get a higher time weight compared to an earlier one, and the RSSI weight should be a linear function of the sample’s RSSI value. This is because samples closer to an AP (i.e., ones with high RSSI values) are known to be more accurate compared to those more distant from the set of APs.

In our experimental evaluation described in Section III, we used the following time and RSSI weight formulas for a positioning sample \( P_i \) received at time \( t_i \) with RSSI value of \( RSSI_i \): \[
    \text{weight}_{\text{time}} = t_{i} - t,
    \]

where \( t \) is the window start time, and \[
    \text{weight}_{\text{RSSI}} = RSSI_i.
\]

We normalized each of these weights as follows: \[
    \text{weight}_{\text{time},n} = \frac{t_{i} - t}{\sum_{n=1}^{N} \text{weight}_{\text{time},n}}
\]

and \[
    \text{weight}_{\text{RSSI},n} = \frac{RSSI_i}{\sum_{n=1}^{N} RSSI_i}.
\]

We illustrate the invocation of the data smoothing algorithm in Section II-D.

### C. The WP High-Level Algorithm

Algorithm 1 is the high-level WP algorithm composed from the noise reduction and data smoothing algorithms described in the previous two sections. The algorithm iterates over the samples (line 3); for each sample, it first checks if it is a noisy one as described in Section II-A(line 5). If the current sample is a valid one, the WP smoothing technique described in Section II-B is applied (line 6), and the calculated WP is saved in the samplesHistory list (line 7).

If the current sample is noisy, it is added to the noisyHistory (line 9) rather than being deleted. This is done from two reasons: i) there is a small probability that this sample is actually a valid one mistakenly classified by the algorithm as a noisy sample; and ii) in a multi-floor venue a person (device) can exceed the “allowed” maximal speed by using the elevator. We note that such a classification error can cause the algorithm to also classify subsequent samples as noisy ones, once a valid sample was eliminated from the data smoothing window. To overcome such cases, the auxiliary function needToSwitch checks if the head of the noisyHistory list is of size noiseThreshold. If so, the samplesHistory list is set to the noisyHistory head and the WP algorithm applies the data smoothing algorithm on the new samplesHistory while emptying the noisyHistory list (lines 11-12).

### D. Handling “Noise Burst”

We next describe an invocation of the WP algorithm on a static device in a lab during a “noise burst”. Before/after this “noise burst” the device is located around coordinates \((25.5, 18.5)\). The exact settings of this experiment are described in the next section. Table I and Table II include the positioning samples before and after the activation of the WP algorithm, respectively. The original first four positioning samples (lines 1–4 in Table I) are valid samples, and hence they are adjusted to reflect the device’s WP (lines 1–4 in Table II). We note these resulted WP samples are closer to the device’s
Table I shows positioning samples of static device located around (25.5, 18.5). Table II shows the output of the WP algorithm on the former input.

actual positioning than the raw (unprocessed) positioning samples generated by the sensing platform. Samples number 5 and 6 are detected as noise, and hence they are eliminated from the samples set and the WP remains unchanged and very close to the device’s actual positioning. In contrast, during this period, the positioning as determined by the sensing platform substantially drifts from the device’s actual positioning. These two trends continue during the entire “noise burst”.

### III. Evaluation

Our empirical evaluation is based on experiments we run in a lab of a big hi-tech company in the United States over the course of 24 hours. This 20x10 meter lab is equipped with a Wi-Fi-based sensing platform from a leading company in this domain. The platform collects RSSI signals from three APs and employs Wi-Fi-based trilateration to infer a device’s positioning. For our experiments we used five mobile devices: four static devices and a single dynamic one. The positioning update frequency was not uniform across all the devices, but average update frequency was about one minute. The experiments took place under lab conditions with close to zero signal interference (e.g., people movement).

We applied the algorithm with $S$ (the speed upper bound, see Section II-A) ranging from 2 kph to 8 kph, advancing in steps of 0.5 kph. This speed parameter mainly affects the probability of detecting noisy samples. Since the experiments were done under the lab conditions described above (with no signal interference in a relatively small lab), the overall percentage of noisy samples was low. Consequently, the speed value in the aforementioned range had a small effect on the overall performance of the algorithms. In all the experiments described below, we set this parameter to be 4.5 kph, as, this represents the average walking speed in a shopping mall [15]. We set the accuracy radius $R$ to 2.5 meters as explained in Section II-A.

#### A. Static Device Test

In this section we present the results from experiments done with four static (mobile) devices. We study the effect of the data smoothing window size on the performances of the algorithm from both noise reduction and data smoothing perspectives. Figure 1 depicts the maximal Euclidean distance (denoted as the noise level) between a pair of positioning samples associated with a given device. Ideally, this distance should be zero for static devices. As the figure shows, without applying any data smoothing algorithms (window size equals to zero), the noise level can be more than twenty meters. As we increase the window size from zero to 60 seconds, there is a monotone decrease of the noise level from 13.5–20.4 meters to 5–8.8 meters, which demonstrates a 75.5% improvement in the data accuracy. Increasing the window size further up to 100 seconds results in a more moderate noise decrease. Hence, following this experiment, we fixed the window size at 60 seconds to achieve the best results for both static and dynamic devices.

![Fig. 1: Noise levels for four static devices.](image-url)

Figure 2 describes the separate contributions of each of the WP’s two sub-algorithms (noise reduction and data smoothing) to the improved data accuracy. We again measured the noise level for each of the aforementioned
static devices. We report the noise levels for i) raw positioning events; ii) raw positioning events that have been processed by the data smoothing algorithm described in Section II-B; and iii) similar to ii) with the additional noise reduction described in Section II-A.

Recall that the experiment took place under lab conditions which as shown in Figure 2 resulted in a low percentage (about 1%) of noise (noisy samples). As indicating our ongoing confidential experiments in retail settings, in real deployments this percentage is substantially higher indicating about the high noise levels induced by existing Wi-Fi-based sensing platforms. However, even under lab conditions it is evident that these platforms are far from accurate, as shown by our experiments. Applying the data smoothing algorithm on raw positioning data resulted in a substantial reduction in the noise level for all four devices: between 3.74 meters / 19.56% for device number 1 and 9.74 meters / 55.45% for device number 2. The additional reduction in the noise level from applying the noise reduction algorithm in addition to the data smoothing one was significant only for device number 4 (4.78 meters / 23.45%) and minor for the other three devices as explained above.

Fig. 2: Effect of the data smoothing and noise reduction algorithms on noise levels

Finally, Figure 3 depicts positioning samples of a static device throughout a 24-hour experiment in the lab. Although the device is static, throughout a long experiment the device’s positioning samples can be scattered across a very big area whose diameter is more than 20 meters. After applying the WP algorithm on this data, the positioning samples are centered around an area that is about 1/4 of the area in which the raw positioning samples are scattered.

B. Dynamic Device Quality Measures

This section describes the performance of the WP algorithm on positioning samples of a dynamic device in the lab. Ideally, these samples should construct relatively straight and smooth trajectories, which represent a person’s walking pattern. Figure 4 describes these samples before and after the application of the WP algorithm with the previously described parameters. As evident, application of the WP algorithm generated a much smoother trajectory that includes substantially less “bouncing”.

To formally quantify the improvement in the smoothness of the generated trajectory following the application of the WP algorithm, we use the following two well-known techniques:

Derivation. For each sequence of samples $C$ (either from the raw data or the smoothed one), we consider their curvature. Revising the notion of derivation, we discretely consider consecutive sample points in our computation. In this sense, we define the first and second derivatives as follows. Let $s_i = (x_i, y_i)$ be any sample and let $v_i = (x_{i+1} - x_i, y_{i+1} - y_i)$ be the vector directed from $s_i$ to $s_{i+1}$. Let $\beta_i = \text{atan}2(y_i, x_i)$ be the angle of $v_i$ with respect to the $x$-axis. The first derivative is defined as the angular difference between two consecutive vectors: $\frac{dC}{dv_{i+1}} = \alpha_{i+1} = \beta_{i+1} - \beta_i$. We also consider the second derivative $\frac{d^2C}{dv_{i+1}} = \alpha_{i+1} - \alpha_i$. The rationale is that these two values measure the smoothness of the curve: the first derivative penalizes turns while the second derivative penalizes the difference in the magnitude of the turns (so walking in a perfect circle will have no penalty). The assumption is that a smooth walking path is preferred over a zigzagged walking path since it simulates a normal walking pattern. In a whole, the smoothing algorithm decreases the first and second derivatives. It is evident in Figure 5 where except from a few locations, the second derivative decreases substantially. However, there are cases where the derivatives increase (as in sample 28 in Figure 5); this is the result of perturbing a short sequence of positions that originally lie on straight lines. Figure 5 depicts the second derivative (as defined above) for the raw and smoothed data. As the figure shows, application of the
WP algorithm substantially decreases the second (and first) order derivatives up to a 10 fold improvement.

**Fitting curves.** We fit low-degree polynomial curves to each trajectory using Gradient Descent or more sophisticated techniques. We also measure the errors of the sampling data with respect to the obtained curves (i.e., their distances from the curves). We omit the details of our experimentation from this report, but it is evident (see Figure 4) that by smoothing the data we decrease the errors significantly.

On the whole, these two methods offer strong evidence to the usefulness of our smoothing algorithm: the significant decrease of the magnitudes we obtained with both methods demonstrates natural movement that is more likely to match reality.

![Fig. 4: Positioning samples of a dynamic device before (left) and after (right) smoothing by the WP algorithm](image)

![Fig. 5: Second derivative (absolute value) results for positioning samples of dynamic device.](image)

**IV. Conclusions**

In this paper we presented a novel WP data cleansing algorithm designed to improve the accuracy of data used in indoor location positioning. Our algorithm can a) detect and remove noisy samples from the positioning samples set; and b) smooth the remaining positioning samples. We also presented the theory behind this algorithm as well as our static and dynamic evaluation. Our experiments showed up to 75.5% improvement in the data accuracy. This improved accuracy opens new market opportunities in the retail and travel and transportation domains.

**References**


