ARTICLE IN PRESS

Pervasive and Mobile Computing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Smart probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices

Igor Bisio, Fabio Lavagetto, Mario Marchese, Andrea Sciarrone *

Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture (DITEN), University of Genoa, Via Opera Pia 13 16145, Genoa, Italy

ARTICLE INFO

Article history:
Received 27 July 2015
Received in revised form 2 February 2016
Accepted 16 February 2016
Available online xxxx

Keywords:
Indoor positioning
WiFi fingerprinting
Computational and energy efficiency
Smartphones

ABSTRACT

Different positioning schemes are based on the probability $p(\mathbf{o}|l)$ to have an observation vector $\mathbf{o}$ at a Reference Point (RP) l, based on Gaussian probabilities. This paper presents an approach to speed-up the $p(\mathbf{o}|l)$ computation without any approximation. The consequent positioning scheme is called *Smart* P-FP. The comparison between *Traditional* (without any $p(\mathbf{o}|l)$ acceleration) and *Smart* P-FP is performed over different smartphones. The saved energy is about 90% for a large number of Access Points (APs) but is significant even with few APs: more than 86% with 3 APs. The proposed $p(\mathbf{o}|l)$ computation is beneficial to any $p(\mathbf{o}|l)$-based positioning scheme.

1. Introduction

In recent years, navigation and localization systems have become extremely popular because positioning data represent a very useful context information for Location Based Services (LBSs). Concerning outdoor positioning, GPS is widely used but, in many environments like urban areas, it is not always available due to the severe attenuation and unpredictable multipath fading that may affect related signals [1]. Indoor positioning techniques are extensively investigated in the literature and different technologies are employed: Bluetooth [2,3], RFID [4,5], Ultra Wide Band (UWB) [6–8], and WLAN. On the other hand, Wireless Local Area Networks (WLANs) based on IEEE 802.11 standard, also called WiFi networks, are often employed for local area and indoor positioning purposes [9] as also done in this paper.

In general, sensing WiFi signals irradiated by Access Points (APs) and measuring specific quantities such as Time Of Arrival (TOA) [10], Time Difference Of Arrival (TDOA) [11], Angle Of Arrival (AOA) [12] and Received Signal Strength (RSS), enable a Mobile Device (MD) e.g., a smartphone, to estimate its position and make such information available for LBSs and Context-Aware (CA) applications. RSS-based localization algorithms have been extensively studied for indoor positioning [13–17]. [18] proposes an algorithm called RADAR: a RF-based system to locate and track users inside buildings by using RSS information gathered at multiple receivers. Compared with other measurement-based algorithms (such as mentioned TOA, TDOA, AOA or Ultra Wide Band), RSS-based approaches can be applied by a WiFi-integrated smartphone without any additional hardware. There are two main approaches for WiFi indoor positioning available in the literature: (i) Multilateration (well known as trilateration, when 3 APs are used) and (ii) FingerPrinting (FP).

The key idea of Multilateration is to estimate the distances between APs and MD by using the strength of the signals transmitted by the APs and received by the MD, to compute an estimation of the mobile device position. [16] employs

* Corresponding author.
E-mail addresses: igor.bisio@unige.it (I. Bisio), fabio.lavagetto@unige.it (F. Lavagetto), mario.marchese@unige.it (M. Marchese), andrea.sciarrone@unige.it (A. Sciarrone).

http://dx.doi.org/10.1016/j.pmcj.2016.02.001
1574-1192/

the Cramer–Rao lower bound to infer the maximum likelihood estimation of the distances; [15] the analytical Triangular Interpolation and eXtrapolation (TIX). [9,19,20] estimate the mentioned distances by using a cubic regressive equation, the Singular Value Decomposition (SVD) or the Least Squares approach.

The main drawbacks of Multilateration are: (i) low localization accuracy and (ii) high energy consumption due to required heavy computations, crucial issue when MDs are involved (see [14,21]).

FingerPrinting (FP) is a two-step procedure. The first step, carried out offline and also known as *training* (offline) phase, is the collection of information aimed at obtaining a spatio-temporal representation of the Received Signal Strengths (RSSs) in a given area. The *training* is based on Reference Points (RPs), usually selected in order to cover the entire area of interest through a uniform grid, and on RSS measurements carried out by an MD and collected for each RP. The set of measured RSSs, at a given RP, is used to build the fingerprint for that RP. Efficient fingerprint building methods are proposed in [20,22–28]. The second step is the *positioning* (online) phase, which starts with an online RSS measurement performed by the MD that wants to determine its position, and is followed by the comparison of the measured RSS with the reference FP of each RP. The estimated position corresponds to the coordinates, or to a combination of the coordinates, of the RPs whose fingerprints match the RSS measures most closely. Examples of fingerprinting employment can be found in [1,9,29].

It is possible to further distinguish between two FP approaches: (a) Deterministic FingerPrinting (D-FP) and (b) Probabilistic FingerPrinting (P-FP) [30]. The former estimates the position by considering only deterministic RSSs measurements [22,23,31] (such as average and variance [9,32,33]). Even if this method provides a reasonable localization accuracy, it ignores much of the information that can be extracted from the *training* data since the RSS, in a given position, can be characterized by more parameters and not only by the simple RSS average value.

P-FP, which is the technique considered as a reference in this paper, computes the position by considering the RSSs measurements as a part of a random process [30] by better exploiting the information present within the acquired signals. It has been applied in [9,13,17,31].

This paper proposes a computational and energy efficient P-FP procedure, suited to be employed over smartphone platforms. The main idea is to structure the process required to get P-FP so as to allow the computation and the storage of some quantities already in the training phase and therefore provide time and energy saving. In more detail, during the positioning phase a mobile device acquires the RSS values and computes an observation vector. Many positioning schemes base their action on the computation of the probability $p(\mathbf{o}|l)$ to have an observation vector $\mathbf{o}$ at a fixed RP l based on Gaussian probabilities. Probabilistic FingerPrint (P-FP) is one of them and is the main comparison benchmark because it computes a Gaussian-based probability $p(\mathbf{o}|l)$ entirely during the online phase. It is identified as *Traditional* P-FP in the following.

This paper introduces a *Smart* $p(\mathbf{o}|l)$ computation scheme by using a reduced number of operations which, consequently, saves time and energy. The positioning scheme, totally derived from *Traditional* P-FP but using the new $p(\mathbf{o}|l)$ computation, is called *Smart* P-FP. It has been practically implemented and tested over off-the-shelf Android OS smartphones. Section 2 contains the state of the art and the key idea of the paper. Section 3 recalls *Traditional* P-FP while Section 4 introduces *Smart* P-FP showing how the factorization of the $p(\mathbf{o}|l)$ equation used in *Traditional* P-FP allows a more efficient and, as a consequence, energetically convenient positioning process without introducing any approximation and consequent accuracy detriment. The obtained results (reported in Section 5) show *Smart* P-FP efficiency when compared with *Traditional* P-FP, in terms of number of operations, time, and energy. Section 6 shows the results when the new $p(\mathbf{o}|l)$ computation procedure is exploited to boost some performing positioning algorithms in the literature based on Gaussian probabilities. Final discussions are reported in Section 7.

2. Related works and key idea of the proposed P-FP approach

[24,34,35] introduce approaches aimed at reducing the computations of the localization algorithms. Other important references for this paper are [30,36], which contain a comparison, carried out by using different devices, among localization methodologies in terms of computational burden and accuracy. Even if P-FP is computationally and energetically lighter than Multilateration, it still requires a not negligible amount of computation and energy resources, in particular for MDs.

The work in [28] presents a probabilistic, WiFi-based, location determination system called *Horus*. It is aimed at satisfying two goals: high accuracy and low computational requirements. Differently from the proposed paper, [28] employs laptops and PCs to acquire RSS measures and to compute the user's position. *Horus* achieves low computational burden by implementing a clustering module. In practice, it defines a cluster as a set of locations (*i.e.,* the RPs) sharing a common set of APs. During the positioning phase, *Horus* searches the RP which better matches the observation vector only within the cluster to which such observation vector belongs. *Horus* shows a great accuracy (with a positioning error below 0.6 (m)). It has direct benefit from the *Smart* $p(\mathbf{o}|l)$ computation as will be shown in the performance evaluation. Even if its main scope is slightly different from the one proposed in this paper, the work in [37] is still an interesting reference to consider. It proposes a zone-based RSS reporting where the location server translates geographical zones defined by LBSs into RSS-based representations. This allows implementing an efficient Place-Of-Interest (POI) detection.

[38] proposes a novel, incremental approach that reduces the energy consumption of WiFi localization by scanning just a few selected channels. This idea provides good energy saving (between 20.64% and 57.79%). It considers channels 1, 6 and 11 as generally preferred by system administrators and sets them as factory defaults. Although this is a reasonable hypothesis, it could raise some issues if such positioning system should be applied to APs which use different channels.

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

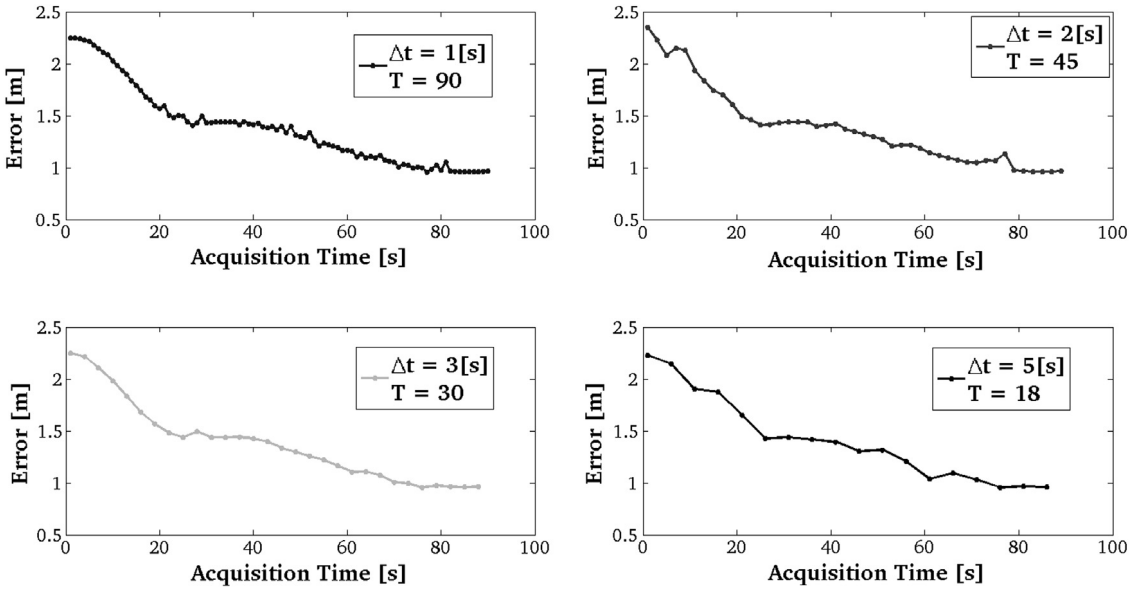


Fig. 1. Positioning error (in (m)) as a function of the acquisition time (in (s)) for different values of T and Δt. The number M of employed APs is 10.

To avoid heavy computational on-board loads, MDs often rely on a web-server component to get their position. In these cases, MD acts as a sensor by acquiring RSS values and sending them to the web-server component which is in charge to perform all the computations necessary to get the position. Even though this approach allows saving MD's CPU, it requires more energy than simple on-board computing because radio transmission is one of the most energy-consuming tasks in mobile devices [39]. As said in [40]: "*for computation offloading to be beneficial the workload needs to perform more than* 1000 *cycles of computation for each byte of data*". Considering that P-FP requires a small number of operations (or cycles of computation), see Section 5.3, transmitting data over the network and using a web-server component are not the optimal choice in this case.

To reduce the number of operations, computation time and energy costs, the key idea of the proposed *Smart* approach is to employ an algebraic factorization of the equations of the P-FP method, which allows computing and storing some necessary variables directly in the *training* phase, thus avoiding their computation during the online *positioning* phase. The key quantity to act on is the Gaussian Probability Density Function (PDF). It is employed in many fields such as statistics (*e.g.*, to describe the Normal distributions), signal processing (*e.g.*, to define Gaussian filters and to model classifiers), and mathematics/physics (*e.g.*, to solve heat and diffusion equations). Many works in the literature are aimed at reducing the computational burden requested by Gaussian-based approaches but, in some cases, approximations are applied and available solutions provide a single simplified closed formula whose numerical results approximate the analytical solution.

In more detail, [41] provides derivations for the mean and standard deviation of a product of two Gaussian PDFs. Even if the idea could be extended to more than two functions, [41] does not provide a way to compute the final formula. [42] contains two algorithms to compute the product of Gaussian mixtures efficiently but it introduces an approximation. Other scientific contributions show similar ideas but are related to sums rather than to products. For example, [43,44] show how the employment of approximations can reduce the number of operations to compute the sum of N Gaussian functions. [45] proposes many ways to practically implement numerical algorithms but does not consider the product of Gaussian PDFs.

3. *Traditional* Probabilistic Fingerprinting approach (*Traditional* P-FP)

Let L and M be the number of employed Reference Points (RPs) and Access Points (APs), respectively. We denote the lth Reference Point with RP_l and the mth Access Point AP_m, where $l \in [1, L]$ and $m \in [1, M]$. Each RP_l is identified by its coordinates $\mathbf{C}_l = (x_l, y_l)$ with respect to a common two-dimensional Cartesian reference system. We define T as the number of RSS *training* values, each of them acquired every Δt seconds (scanning period). Many different values of the parameters T and Δt (defined for the *training* phase) have been tested: too large values lead to long acquisition time; too small values imply getting less measurements and consequently unreliable observation vectors.

Fig. 1 shows the trends of the positioning error (in (m)) as a function of the acquisition time (in (s)) for different values of T and Δt when $M = 10$ APs are employed. The overall acquisition time considered is 90 (s). When $\Delta t = 1$ (s) the whole positioning process collects $T = 90$ RSS values. Similarly, when $\Delta t = 2$ (s) it acquires $T = 45$ RSS values and so on. From Fig. 1 it is possible to note a decreasing trend of the positioning error with respect to the increase of the acquisition time. This is motivated by the fact that a longer acquisition time provides a more robust average value of the RSS and, as a consequence, more reliable observation vectors.

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

During the *training* phase, a three-dimensional observation matrix $\tilde{\mathbf{O}}$ composed of L rows and M columns of T elements is built. The element $\tilde{o}_{m,l,t}$, $m \in [1, M]$, $l \in [1, L]$, $t \in [1, T]$ of the observation matrix $\tilde{\mathbf{O}}$ is the single RSS value received from the mth AP, sensed at the lth RP during the tth signal strength measure.

Fixing m and l and varying t from 1 to T in the observation matrix, we get a single observation vector composed of T RSSs values. Every measurement is made by keeping a MD still in a specific *RP* and by varying its orientation. $\mu_{m,l}$ and $\sigma^2_{m,l}$ are the mean and the variance of all the observations for the mth AP at the lth RP, respectively. They represent the radio fingerprint of each RP.

$$\begin{cases} \mu_{m,l} = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} \tilde{o}_{m,l,t} \\ \sigma^2_{m,l} = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} \left(\tilde{o}_{m,l,t} - \mu_{m,l}\right)^2 . \end{cases} \tag{1}$$

The procedure is iterated for all the considered RPs. Coherently with the literature (e.g., [31,17,46]), a Gaussian Probability Density Function (PDF), is used to model the RSSs distribution in this paper. In more detail, $N(\cdot)$ indicates the one-dimensional Gaussian PDF that models the RSS variation for a single AP at the single RP_l. During the offline phase, we estimate an M-dimensional Joint Gaussian Probability Density Function (JG-PDF) modelling the RSSs variations for all the APs for each RP in the area of interest. This is necessary because the RSS measured at a generic RP is related to the radio signals transmitted by all the M APs and not by one single AP. In order to estimate such PDF, coherently with the literature, we apply the hypothesis of statistical independence among the irradiated RSSs from each AP. This assumption implies that the M-dimensional JG-PDF can be expressed as the product of M mono-dimensional Gaussian PDFs:

$$N_M\left(\mu_l, S_l\right) = \prod_{m=1}^{M} N\left(\mu_{m,l}, \sigma^2_{m,l}\right) \tag{2}$$

where $N_M(\cdot)$ represents the M-dimensional JG-PDF that describes the RSS distributions for all the APs at the specific RP_l. The quantities μ_l and S_l are, respectively, the means vectors ($1 \times M$) and the covariance matrices ($M \times M$) computed at the RP_l. At the end of the *training* phase a probabilistic radio map based on the distribution of the received RSSs is obtained for the area of interest. The single PDF for each AP_m at each RP_l can be written as:

$$N\left(\mu_{m,l}, \sigma^2_{m,l}\right) = \frac{1}{\sqrt{2\pi\sigma^2_{m,l}}} e^{-\frac{\left(o_m - \mu_{m,l}\right)^2}{2\sigma^2_{m,l}}} . \tag{3}$$

During the positioning phase the MD acquires the RSS values and computes the observation vector $\mathbf{o} = [o_1, \ldots, o_m, \ldots, o_M]$ similarly as done for the *training* phase. The positioning (online) phase checks the Gaussian PDF $N(\cdot)$ value for the observation vector $\mathbf{o} = [o_1, \ldots, o_m, \ldots, o_M]$ and computes the quantity $p(l|\mathbf{o})$ by using the Bayes rule as in the following formula:

$$p(l|\mathbf{o}) = \frac{p(\mathbf{o}|l)p(l)}{p(\mathbf{o})}. \tag{4}$$

Given $p(l)$ and $p(o)$, $p(l|\mathbf{o})$ only depends on the probability $p(\mathbf{o}|l)$ to have $\mathbf{o}$ at a given RP_l. In P-FP, the values of $p(\mathbf{o}|l)$ are computed as in Eq. (5) by combining Eqs. (2) and (3):

$$p(\mathbf{o}|l) = \prod_{m=1}^{M} \frac{1}{\sqrt{2\pi\sigma^2_{m,l}}} e^{-\frac{\left(o_m - \mu_{m,l}\right)^2}{2\sigma^2_{m,l}}}, \quad \forall l \in [1, L] . \tag{5}$$

Eq. (5) reports the reference formula for the *Traditional* P-FP. To obtain a fair comparison of *Traditional* P-FP with the proposed *Smart* P-FP, all the results which involved the *Traditional* approach have been obtained by applying Eq. (5) as is, without any pre-manipulations.

The estimated coordinates of the Mobile Device (MD) position, denoted with $\tilde{\mathbf{C}} = (\tilde{x}, \tilde{y})$, may be computed by using different criteria. Among others, we can mention: Most Likelihood (ML) which simply provides, as MD position, the Cartesian coordinates of the RP_l with the highest $p(\mathbf{o}|l)$; K-ML with $K \geq 2$, which averages the coordinates of the K RPs with the highest $p(\mathbf{o}|l)$ values; K Weighted Most Likelihood (KW-ML) with $K \geq 2$, which computes a weighted average of the coordinates of the K RPs with the highest $p(\mathbf{o}|l)$ values.

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

4. The Smart P-FP approach

Smart P-FP procedure is based on the following algebraic steps. Eq. (5) may be rewritten as follows:

$$p(\mathbf{o}|l) = \prod_{m=1}^{M} \frac{1}{\sqrt{2\pi\sigma_{m,l}^2}} e^{-\frac{(o_m-\mu_{m,l})^2}{2\sigma_{m,l}^2}} = \frac{1}{\sqrt{2\pi\sigma_{1,l}^2}} e^{-\frac{(o_1-\mu_{1,l})^2}{2\sigma_{1,l}^2}} \cdot \frac{1}{\sqrt{2\pi\sigma_{2,l}^2}} e^{-\frac{(o_2-\mu_{2,l})^2}{2\sigma_{2,l}^2}} \cdots \frac{1}{\sqrt{2\pi\sigma_{M,l}^2}} e^{-\frac{(o_M-\mu_{M,l})^2}{2\sigma_{M,l}^2}}. \tag{6}$$

We define the quantity $\bar{\sigma}_{m,l}^2$ as:

$$\bar{\sigma}_{m,l}^2 = \sigma_{1,l}^2 \cdots \sigma_{m-1,l}^2 \cdot \sigma_{m+1,l}^2 \cdots \sigma_{M,l}^2 = \frac{\prod_{m=1}^{M} \sigma_{m,l}^2}{\sigma_{m,l}^2}. \tag{7}$$

After some algebraic computations, we get:

$$p(\mathbf{o}|l) = \left(\frac{1}{\sqrt{2\pi}}\right)^M \frac{1}{\sqrt{\prod_{m=1}^{M} \sigma_{m,l}^2}} e^{-\left[\frac{\sum_{m=1}^{M} \bar{\sigma}_{m,l}^2 (o_m-\mu_{m,l})^2}{2\prod_{m=1}^{M} \sigma_{m,l}^2}\right]}. \tag{8}$$

Considering the numerator of the exponential power, we can write:

$$\begin{aligned} \sum_{m=1}^{M} \bar{\sigma}_{m,l}^2 (o_m-\mu_{m,l})^2 &= \sum_{m=1}^{M} \frac{\prod_{j=1}^{M} \sigma_{j,l}^2}{\sigma_{m,l}^2} (o_m-\mu_{m,l})^2 \\ &= \sum_{m=1}^{M} \frac{\prod_{j=1}^{M} \sigma_{j,l}^2}{\sigma_{m,l}^2} \left(o_m^2 + \mu_{m,l}^2 - 2o_m\mu_{m,l}\right) \\ &= \sum_{m=1}^{M} o_m^2 \frac{\prod_{j=1}^{M} \sigma_{j,l}^2}{\sigma_{m,l}^2} + \mu_{m,l}^2 \frac{\prod_{j=1}^{M} \sigma_{j,l}^2}{\sigma_{m,l}^2} - 2o_m\mu_{m,l} \frac{\prod_{j=1}^{M} \sigma_{j,l}^2}{\sigma_{m,l}^2}. \end{aligned} \tag{9}$$

It is worth noting that quantities $\frac{\prod_{j=1}^{M} \sigma_{j,l}^2}{\sigma_{m,l}^2}$, $\mu_{m,l}$, and $\mu_{m,l}^2$, $\forall m$, do not contain terms acquired in the *positioning* phase. Consequently, they can be computed and stored during the *training* phase so saving operations, time and energy. Starting from Eq. (9), we can also define three quantities that will be useful in the following:

$$\begin{cases} \alpha_l = \dfrac{\gamma_l}{2} \cdot \displaystyle\sum_{m=1}^{M} \frac{\mu_{m,l}^2}{\sigma_{m,l}^2} \\ \beta_l = \left(\dfrac{1}{\sqrt{2\pi}}\right)^M \cdot \dfrac{1}{\sqrt{\frac{\gamma_l}{2}}} \\ \gamma_l = \displaystyle\prod_{m=1}^{M} \sigma_{m,l}^2 \end{cases} \tag{10}$$

α_l, β_l and γ_l are the elements of the following three vectors of length L: $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_l, \ldots, \alpha_L]$, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_l, \ldots, \beta_L]$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_l, \ldots, \gamma_L]$, respectively. Since $\mu_{m,l}^2$ and $\sigma_{m,l}^2$ are already known after the *training* phase, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ can be computed without the knowledge of the observation vector. Eq. (9) may be rewritten by using α_l and γ_l as follows:

$$\sum_{m=1}^{M} o_m^2 \frac{\prod_{j=1}^{M} \sigma_{j,l}^2}{\sigma_{m,l}^2} + \mu_{m,l}^2 \frac{\prod_{j=1}^{M} \sigma_{j,l}^2}{\sigma_{m,l}^2} - 2o_m\mu_{m,l} \frac{\prod_{j=1}^{M} \sigma_{j,l}^2}{\sigma_{m,l}^2} = \alpha_l + \frac{\gamma_l}{2} \sum_{m=1}^{M} \frac{o_m(o_m - 2\mu_{m,l}^2)}{\sigma_{m,l}^2}. \tag{11}$$

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

Table 1
Technical features of the three mobile devices employed in this paper.

	Device 1	Device 2	Device 3
CPU	Cortex-A8 1 GHz	MSM8255T 1.4 GHz	ARM 11 528 MHz
Memory	8 GB/16 GB 512 MB RAM 2 GB ROM	4 GB 512 MB RAM 512 MB ROM	288 MB 512 MB RAM 512 MB ROM
Wi-Fi	802.11 b/g/n	802.11 b/g/n	802.11 b/g
OS	Android v2.1 (Eclair)	Android v2.3 (GingerBread)	Android v1.6 (Donut)

The quantity $\sum_{m=1}^{M} \frac{o_m(o_m-2\mu_{m,l}^2)}{\sigma_{m,l}^2}$ is the only part of Eq. (11) that must be computed during the positioning (online) phase. Taking into account the expressions in Eq. (10), the exponential power (ignoring the sign) of Eq. (8) can be written as:

$$\frac{\sum_{m=1}^{M} \bar{\sigma}_{m,l}^2 (o_m - \mu_{m,l})^2}{2\prod_{m=1}^{M} \sigma_{m,l}^2} = \frac{\alpha_l + \frac{\gamma_l}{2} \sum_{m=1}^{M} \frac{o_m(o_m-2\mu_{m,l}^2)}{\sigma_{m,l}^2}}{\gamma_l}$$

$$= \sum_{m=1}^{M} \frac{\mu_{m,l}^2 + o_m(o_m - 2\mu_{m,l}^2)}{2\sigma_{m,l}^2}. \tag{12}$$

Consequently, the $p(\mathbf{o}|l)$ probability in Eq. (8) can be rewritten as

$$p(\mathbf{o}|l) = \beta_l e^{-\sum_{m=1}^{M} \frac{\mu_{m,l}^2 + o_m(o_m-2\mu_{m,l}^2)}{2\sigma_{m,l}^2}} \tag{13}$$

where the quantities $\mu_{m,l}^2$, $2\mu_{m,l}$, $2\sigma_{m,l}^2$ and β_l can be pre-computed and stored in the *training* phase. The computation in Eq. (13) is called *Smart*. It is important pointing out that the result reported in Eq. (13) does not introduce any approximation at all. This fact leads to a new P-FP method, called *Smart* P-FP, that allows saving time and energy (as deeply detailed in the next sections) without incurring in higher positioning errors. It is important to remind that any positioning method using a Gaussian-based $p(\mathbf{o}|l)$ can have benefit from the application of Eq. (13).

5. Performance evaluation: comparison of *Traditional* and *Smart* approach

5.1. Employed devices

The first part of performance evaluation proposed in this paper has been carried out by implementing *Traditional* and *Smart* P-FP methods on three off-the-shelf MDs. In particular, we have employed three smartphones identified as Device 1, Device 2 and Device 3 and, for each of them, we have computed $p(\mathbf{o}|l)$ in different conditions. All used smartphones are based on the Android OS. Table 1 reports their main technical details.

5.2. Basic information

In this section we analyse the improvement, in terms of number of operations and energy consumption assured by the *Smart* P-FP approach in comparison with the *Traditional* one. In more detail, we compare the mentioned metrics to compute Eq. (5) (*Traditional*) and Eq. (13) (*Smart*). The computation of $p(\mathbf{o}|l)$ is the only differentiation between *Traditional* and *Smart* and, consequently, it is the only discrimination element.

Similarly to [47,48], we adopt the necessary number of FLoating point OPerations (FLOPs[1]) to compare *Traditional* and *Smart* approaches in terms of computational load. Although FLOP counting does not characterize the computational complexity, it is a relevant measure of the computation load [47]. This idea is also supported by [49] which states that FLOPs are a good measure of how fast a computer can work in an ideal situation. Furthermore, it adds: "*since a lot of scientific code mostly deals with floating floats, FLOPs are a reasonable measure of how fast the computer can do meaningful computations*".

In addition, considering that some papers such as [21,50] use the energy consumption to evaluate applications, algorithms and procedures, we have evaluated the two $p(\mathbf{o}|l)$ computation methods also from the energy/power consumption

[1] The acronym FLOP is a flop count, i.e., a count of these operations required by a given algorithm or computer program. It must not be confused with the acronym FLOPS (with the final capital "S") which stands for FLoating-point Operations Per Second, which is a measure of computer performance.

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

Table 2
Estimated times (in (ns)) needed to perform P-FP operations for each employed device.

	T_{sum}	T_{mult}	T_{pow}	T_{exp}	T_{sqrt}
Device 1	25.27	73.35	296.61	110.26	695.72
Device 2	28.08	17.21	208.25	57.76	354.03
Device 3	665	561	4433	4462	10 407

Table 3
Estimated FLOPs.

	F_{sum}	F_{mult}	F_{pow}	F_{exp}	F_{sqrt}
Device 1	1	2.9	11.7	4.3	27.5
Device 2	1	0.6	7.4	2.05	12.6
Device 3	1	0.8	6.6	6.7	15.6

viewpoint. This aspect is crucial for mobile devices, especially smartphones, which are powered by batteries limited in capacity.

In order to estimate the required number of FLOPs, a relation between the single arithmetical operation and the related number of FLOPs is needed [51]. To get this relation, we have considered the algebraical sum as a reference and we have measured the CPU time needed to execute the sum, denoted with T_{sum}. The CPU time required to perform all other operations necessary to compute Eqs. (5) and (13) may be derived starting from T_{sum}. In practice, assuming the number of FLOPs of a single sum, denoted with F_{sum}, equal to 1, the number of FLOPs necessary for any other operation is the ratio between the measured time spent to compute it and T_{sum}.

Table 2 reports the measured time required to perform all the basic operations essential to compute $p(\mathbf{o}|l)$ in the *Traditional* and/or *Smart* FP approach: sum (T_{sum}), multiplication and ratio (T_{mult}), power (T_{pow}), exponential (T_{exp}) and square root (T_{sqrt}). The shown values are the averages, in (ns), obtained by running each single operation for 10^6 times over the three employed devices.

Table 3 reports the obtained estimated FLOPs for each operation.

Both Tables 2 and 3 evidence significant differences both among the devices performing the same operation (*i.e.*, fixing the column and varying the row) and among different operations performed by the same device (*i.e.*, fixing the row and varying the column).

5.3. Smart vs. Traditional P-FP: computational load in the positioning (online) phase

The required computational load during the positioning (online) phase for *Traditional* and *Smart* P-FP approaches can be analytically expressed by considering the overall number of operations in both cases, given the number M of APs for each device.

Concerning *Traditional* P-FP, the computation of the probability $p(\mathbf{o}|l)$ in Eq. (5) for a single RP_l requires:

- $N^T_{sum} = M$ sums
- $N^T_{mult} = 7M - 1$ multiplications
- $N^T_{pow} = 3M$ powers
- $N^T_{exp} = M$ exponentials
- $N^T_{sqrt} = M$ square roots.

In terms of estimate FLOPs Ω_T:

$$\Omega_T(M) = F_{sum}N^T_{sum} + F_{mult}N^T_{mult} + F_{pow}N^T_{pow} + F_{exp}N^T_{exp} + F_{sqrt}N^T_{sqrt}. \tag{14}$$

The time, denoted with $T_T(M)$, needed to compute $p(\mathbf{o}|l)$ can be written as follows:

$$T_T(M) = T_{sum}N^T_{sum} + T_{mult}N^T_{mult} + T_{pow}N^T_{pow} + T_{exp}N^T_{exp} + T_{sqrt}N^T_{sqrt}. \tag{15}$$

Substituting the numerical values reported in Table 3 in Eq. (14) and the numerical values appearing in Table 2 in Eq. (15), it is possible to explicitly write the number of floating point operations and the time required by the *Traditional* approach to compute $p(\mathbf{o}|l)$ for each employed device as a function of the number M of used APs. The obtained functions are reported in Table 4.

Concerning *Smart* P-FP procedure where $p(\mathbf{o}|l)$ is computed through Eq. (13): as said in Section 4, since the quantities $\mu^2_{m,l}$, $2\mu_{m,l}$, $2\sigma^2_{m,l}$ and β_l in Eq. (13) can be pre-computed and stored during the *training* phase, a significant amount of operations and time can be saved during the online (positioning) phase. So, to compute the probability $p(\mathbf{o}|l)$ in Eq. (13), *Smart* P-FP requires:

Table 4
Estimated computational load in the online phase for each employed device, *Traditional* procedure.

Employed device	$\Omega_T(M)$	$T_T(M)$ (ns)
Device 1	$88.8M - 2.9$	$2243M - 73.28$
Device 2	$42.05M - 0.6$	$1180.76M - 16.85$
Device 3	$48.7M - 0.8$	$32385.5M - 532$

Table 5
Estimated computational load in the online phase for each employed device, *Smart* procedure.

Employed device	$\Omega_S(M)$	$T_S(M)$ (ns)
Device 1	$8.8M + 6.2$	$222M + 156.7$
Device 2	$4.2M + 1.65$	$118M + 46.3$
Device 3	$4.6M + 6.5$	$3059M + 4322$

- $N^S_{sum} = 3M - 1$ sums
- $N^S_{mult} = 2M + 1$ multiplications
- $N^S_{exp} = 1$ exponential.

The consequent number of estimated FLOPs and needed time as a function of M are contained in Eqs. (16) and (17), respectively. Substituting the values of Tables 3 and 2 in Eqs. (16) and (17), respectively, we can numerically write the number of FLOPs and time required by *Smart* P-FP procedure to compute $p(\mathbf{o}|l)$ as a function of the number of M APs. Functions are shown in Table 5. Specifically, Table 5 does not consider the pre-computed calculations required by the *Smart* P-FP, but only reports the quantities related to the online computation.

$$\Omega_S(M) = F_{sum}N^S_{sum} + F_{mult}N^S_{mult} + F_{exp}N^S_{exp} \tag{16}$$

$$T_S(M) = T_{sum}N^S_{sum} + T_{mult}N^S_{mult} + T_{exp}N^S_{exp}. \tag{17}$$

5.3.1. Is offload computation beneficial for Smart and Traditional P-FP?

The functions appearing in Tables 4 and 5 will be exploited in the next Sections to focus on the advantages brought by the *Smart* P-FP procedure but they are useful also to comment about the fact that smartphones often rely on a web-server or cloud component to locate the user inside an indoor area so to avoid heavy computations locally, as said at the beginning of Section 2. This action allows saving smartphone's CPU, but may require more energy than executing all the calculations locally since radio transmission is an energy-consuming task [39]. As reported in [40], a good thumb rule to understand if the offload computation can be beneficial is to evaluate if the workload requires more than 1000 Cycles of Computation (CC) for each byte of data.

Since most modern microprocessors can perform $4\left[\frac{\text{FLOPs}}{\text{CC}}\right]$ [52], more than $4\left[\frac{\text{FLOPs}}{\text{CC}}\right] * 1000\left[\frac{\text{CC}}{\text{byte}}\right] = 4000\left[\frac{\text{FLOPs}}{\text{byte}}\right]$ are necessary to make offload computation convenient. Typically, a single float variable is represented and stored by 4 bytes. Both *Traditional* and *Smart* P-FP approaches have to compute $p(\mathbf{o}|l)$ (see Eqs. (5) and (13), respectively). Being the vector $\mathbf{o}$ composed of M float values, both approaches must elaborate $4M$ bytes during the positioning phase.

As reported in Tables 4 and 5, *Traditional* and *Smart* P-FP approaches require, referring to the slowest device, $(88.8M - 2.9)$ and $(8.8M + 6.2)$ FLOPs, respectively. Consequently, *Traditional* P-FP needs $\frac{88.8M-2.9}{4M}\left[\frac{\text{FLOPs}}{\text{byte}}\right]$ ranging from 21.84 to $22.16\left[\frac{\text{FLOPs}}{\text{byte}}\right]$ for $2 \leq M \leq 20$. In the same way, *Smart* P-FP method requires $\frac{8.8M+6.2}{4M}\left[\frac{\text{FLOPs}}{\text{byte}}\right]$ ranging from 2.98 to $2.28\left[\frac{\text{FLOPs}}{\text{byte}}\right]$ for $2 \leq M \leq 20$. For the sake of completeness, it is important to notice that such values, either for *Traditional* and *Smart* approach, refer to the computation load (in $\left[\frac{\text{FLOPs}}{\text{byte}}\right]$) when a single RP is considered. For actual location estimation, the same computation must be performed for every single RP so as to be able to determine which is the most likely location. As a consequence, the minimum number of RPs for the offload computation to be beneficial can be determined for both approaches. The *Traditional* approach (for the worst case, $M = 20$) would require $\frac{4000\left[\frac{\text{FLOPs}}{\text{byte}}\right]}{22.16\left[\frac{\text{FLOPs}}{\text{byte}}\right]}\text{RP} \approx 181$ RPs. In the same way, the *Smart* procedure would require $\frac{4000\left[\frac{\text{FLOPs}}{\text{byte}}\right]}{2.98\left[\frac{\text{FLOPs}}{\text{byte}}\right]}\text{RP} \approx 1342$ RPs. From the obtained numbers it is clear that, for the *Traditional* approach, the offloading computation becomes effective when more than 181 RPs are employed, which is an often verified condition. On the other hand, for the *Smart* procedure more than 1342 RPs should be employed for offloading computation

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

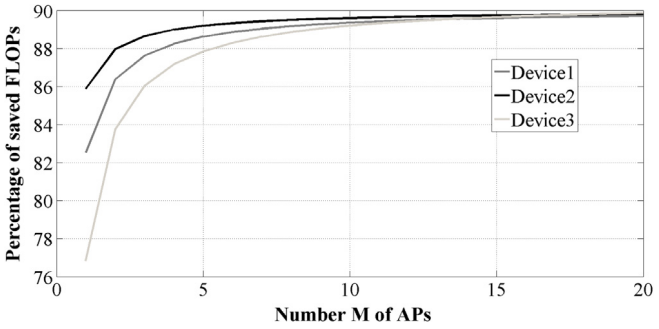


Fig. 2. Estimated percentage of saved number of FLOPs by the *Smart* procedure vs. the number M of APs for each employed device.

Table 6
Percentage of FLOPs saved by the *Smart* procedure as a function of M and limit for $M \to \infty$.

Devices	$\Psi_{\Omega}(M)$	$\lim_{M\to\infty} \Psi_{\Omega}(M)$
Device 1	$\left(1 - \frac{8.8M+6.2}{88.8M-2.9}\right) \cdot 100$	90.09
Device 2	$\left(1 - \frac{4.2M+1.65}{42.05M-0.6}\right) \cdot 100$	90.01
Device 3	$\left(1 - \frac{4.6M+6.5}{48.7M-0.8}\right) \cdot 100$	90.55

to be beneficial. Considering offloading computation requires a reliable and always-guaranteed internet connection, which could be a not always verified hypothesis, if a scenario with more than 181 RPs is considered, the employment of the *Smart* procedure guarantees significant energy savings without using offload computation.

5.4. Comparison of Smart and Traditional P-FP

Traditional and *Smart* P-FP are compared in terms of (i) number of FLOPs and time and (ii) energy/power consumption.

5.4.1. Number of FLOPs and time

Eq. (18) estimates the percentage $\Psi_{\Omega}(M)$ of saved number of FLOPs by the *Smart* procedure as a function of the number M of used APs to compute $p(\mathbf{o}|l)$:

$$\Psi_{\Omega}(M) = \left(1 - \frac{\Omega_S(M)}{\Omega_T(M)}\right) \cdot 100. \tag{18}$$

Taking the values of $\Omega_T(M)$ and $\Omega_S(M)$ in Tables 4 and 5, respectively, Table 6 shows the expression of Eq. (18) for each employed device as a function of M, together with the limit of Eq. (18) for $M \to \infty$. The behaviour versus M of the functions $\Psi_{\Omega}(M)$ appearing in Table 6 is shown in Fig. 2 for each device.

The *Smart* procedure allows saving more FLOPs when the APs number increases. This is important because the higher the AP number, the higher the accuracy of the positioning. It is important also to note that the behaviour is very similar for all employed devices and that the convergence towards the asymptote (approximately 90% for all devices) is quick: if $M \geq 8$ there is no meaningful difference among the percentage results provided by the employed devices because all curves are very close to the asymptote. Another important aspect is that the number of saved operations is significant even if few APs are available: the employment of $M = 3$ APs already provides a percentage of saved operations of 87.6% for Device 1, of 88.64% for Device 2, and of 86.03% for Device 3.

The percentage $\Psi_T(M)$ of saved time by the *Smart* procedure as a function of M to compute $p(\mathbf{o}|l)$ is shown in Eq. (19). Taking the values of $T_T(M)$ and $T_S(M)$ appearing in Tables 4 and 5, respectively, Table 7 shows the expression of Eq. (19) for each employed device and the limit of Eq. (19) for $M \to \infty$. Analytically:

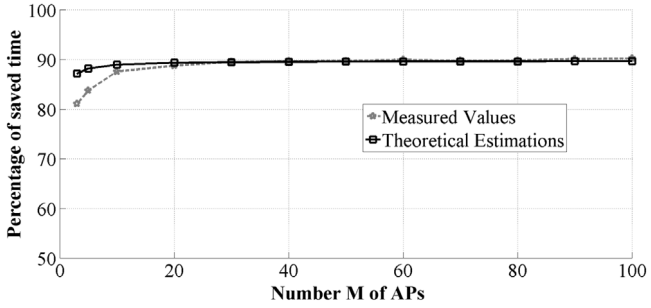
$$\Psi_T(M) = \left(1 - \frac{T_S(M)}{T_T(M)}\right) \cdot 100. \tag{19}$$

Functions $\Psi_T(M)$ in Table 7 are shown versus M in Fig. 3 for each device. As commented for the number of FLOPs, the estimated percentage of saved terms rapidly approaches the asymptote when M increases and, even for very small M values, it is well above 85% for any device. A great advantage to use estimations in Eqs. (18) and (19) is the chance to analyse the behaviour when M is very big (up to $M \to \infty$). This is not possible by performing real measures that, on the other hand, can be carried out when M is not huge and also compared with the values obtained by the estimations.

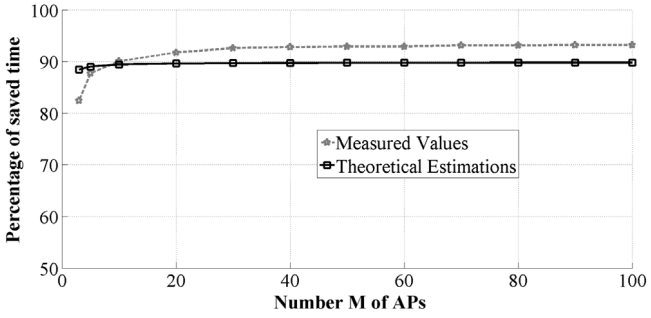
Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

Table 7
Percentage of time saved by the *Smart* procedure as a function of M and limit for $M \to \infty$.

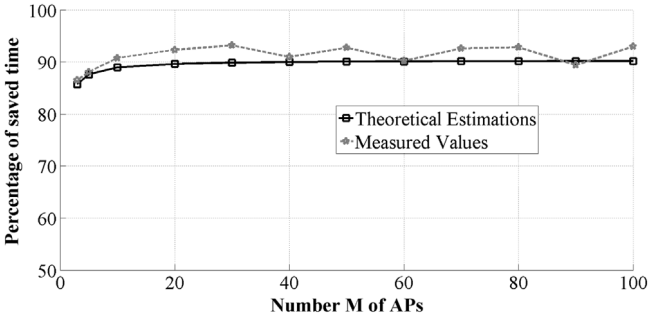
Devices	$\Psi_T(M)$	$\lim_{M\to\infty} \Psi_T(M)$
Device 1	$\left(1 - \frac{222M+156.7}{2243M-73.28}\right) \cdot 100$	90.09
Device 2	$\left(1 - \frac{118M+46.3}{1180,76M-16.85}\right) \cdot 100$	90.01
Device 3	$\left(1 - \frac{3059M+4322.5}{32385.5M-532}\right) \cdot 100$	90.55



(a) Percentage of saved time for Device 1.



(b) Percentage of saved time for Device 2.



(c) Percentage of saved time for Device 3.

Fig. 3. Percentage of saved time assured by the *Smart* procedure with respect to the *Traditional* one for Devices 1 (a), 2 (b) and 3 (c): practical measures compared with estimations.

Practical measurements in terms of saved time (the number of FLOPs are not directly measurable) have been carried out on real smartphones. The time necessary to compute $p(\mathbf{o}|l)$ through *Traditional* and *Smart* P-FP has been measured by varying the number of employed APs. The Received Signal Strength (RSS) of the APs involved in the localization process has been simulated to simplify the implementation of the tests. In order to obtain robust and significant results, all the tests have been performed 10^8 times[2] and then averaged.

[2] Since the time necessary to carry out a single positioning process is very small (*i.e.*, few nanoseconds), the number of tests has been increased in order to obtain measurable quantities.

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

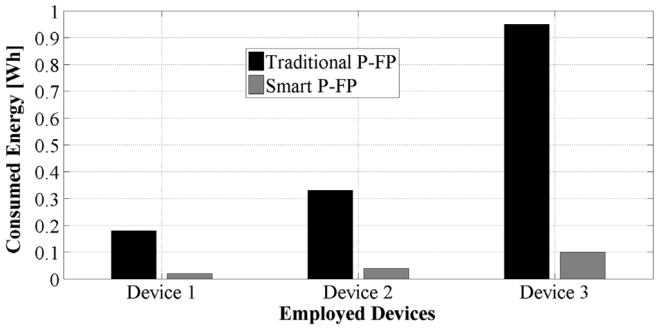


Fig. 4. Comparison of *Smart* and *Traditional* procedure concerning the average consumed energy of each employed device. M varies cyclically in the range [3, 100] for each run.

Table 8
Consumed power as a percentage of the Total Charge and available TCs.

Device employed	*Traditional P-FP*	*Smart P-FP*	*TC* (Wh)
Device 1	3.08%	0.33%	5.55
Device 2	5.50%	0.54%	5.55
Device 3	18.7%	1.94%	4.95

The percentage of saved time guaranteed by the *Smart* procedure with respect to the *Traditional* one when measures on real smartphones are performed is shown in Fig. 3 for all the employed devices versus the number of employed APs. The estimated values already shown in Fig. 2 are reported again in Fig. 3 to allow an immediate comparison.

The theoretical estimation behaviour approximates the real measure very well, so the observations and comments previously reported for Figs. 2 and 3 take an immediate practical meaning.

5.4.2. Energy consumption

Since MDs, especially smartphones, are powered with batteries which are limited in capacity [53] and since relying on web servers and cloud computing is not convenient, energy consumption assumes a special interest. We have measured the energy consumed by *Smart* and *Traditional* P-FP for each employed device by using Android PowerTutor (PT) tool. PT is a power estimation software implemented for Android platform smartphones, which provides accurate, real-time power consumption estimates for power-intensive hardware components including CPU, LCD display, GPS, WiFi, audio, and radio interfaces. In terms of accuracy, for a 10 (s) interval, the average error provided by PT, with respect to measures obtained with hardware metres, is 2.5% [54]. In this paper, we consider the consumed energy expressed in Watt hour (Wh) (3600 (J)), as the product between the power needed to complete the positioning phase and the related processing time. We also compare the necessary energy with the Total Charge (TC) (Wh) of the smartphones batteries. Fig. 4 shows the consumed energy by each device to compute $p(\mathbf{o}|l)$ for *Smart* and *Traditional* P-FP. Reported values are averaged over 10^8 positioning runs because the energy of a single or few positioning runs is not detectable by the PT tool. The same averaged values may be obtained through 10^6, 10^7 or more positioning runs. The used number of APs varies cyclically for each run from 3 to 100. The effect, given the number of performed run, is very similar to a random selection of the number of APs in the range [3, 100]. Again, RSS is simulated but the power is measured on real devices.

The energy saved by the *Smart* approach (grey bars) is meaningful with respect to the *Traditional* one (black bars). Device 1 consumes almost 0.2 (Wh) when *Traditional* P-FP approach is used while it requires slightly more than 0.01 (Wh) if *Smart* P-FP is applied. Device 2 uses 0.3 (Wh) and 0.03 (Wh) for *Traditional* and *Smart* P-FP, respectively. Device 3 consumes 0.96 (Wh) for *Traditional* P-FP and 0.09 (Wh) for *Smart* P-FP. The percentage gain assured by *Smart* P-FP with respect to *Traditional* P-FP is above or equal to 90% for all employed devices. The consumed energy has a direct impact on the smartphone battery charge. The average consumed energy to compute $p(\mathbf{o}|l)$ reported in Fig. 4 is shown in Table 8 as a percentage of the Total Charge (TC) for each device. Table 8 contains also the value of the TC in (Wh) of each employed smartphone. The benefits provided by the *Smart* procedure are very clear: on each device, to compute $p(\mathbf{o}|l)$, if the *Traditional* method consumes x% of TC, the *Smart* one consumes approximately $0.1 \cdot x$% of TC.

5.5. Accuracy of the position and power consumption to compute $p(\mathbf{o}|l)$

This section is aimed at discussing the relation between positioning error (expressed in (m)) and required power (in (mW)) to compute $p(\mathbf{o}|l)$ for the two approaches. Fig. 5 contains positioning error (computed through the 4-WML algorithm with 156 RPs) and consumed power to compute $p(\mathbf{o}|l)$ versus the number M of employed APs for *Smart* and *Traditional* procedure, for Device 2. Devices 1 and 3 provide similar results. It may be noted that Fig. 5 reports the power needed to compute $p(\mathbf{o}|l)$ and not to complete the overall localization process. This is motivated, as already said, by the fact that $p(\mathbf{o}|l)$

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

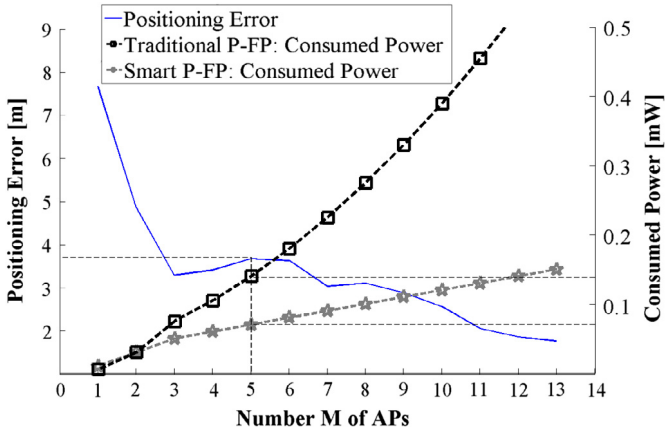


Fig. 5. Positioning error and power consumption versus the number M of used APs.

is the only discriminant factor between *Smart* and *Traditional* P-FP. Anyway, for the sake of completeness, it is meaningful to say that the computation of $p(\mathbf{o}|l)$ within 4-WML with 156 RPs takes most of the computation time: 93.3% of the overall positioning time for *Smart* and 95.92% for *Traditional*.

The two methods provide the same positioning error, since *Smart* P-FP does not introduce any approximation. Fixing a given positioning error or, equivalently, the number of APs, we can get the power that must be applied to compute $p(\mathbf{o}|l)$ for the two approaches directly in Fig. 5. For example, if we want a positioning error below 3 (m) we need to use at least 7 APs. The consequent required power is about 0.1 (mW) for the *Smart* solution and slightly below 0.25 (mW) for *Traditional* one. The advantage in the employment of the *Smart* solution can be also highlighted by choosing a power threshold to compute $p(\mathbf{o}|l)$ and checking the positioning accuracy for the two approaches. Let the maximum amount of power available be, for example, 0.15 (mW): if the *Traditional* approach is employed, no more than 5 APs can be used and the positioning error is around 4 (m); if *Smart* P-FP is applied, 12 APs can be used so allowing a positioning error below 2 (m).

5.6. Power consumption distribution between WiFi interface and CPU

This section is aimed at evaluating the distribution of the energy consumption to compute $p(\mathbf{o}|l)$ between the two hardware components involved in the positioning process: WiFi interface (to scan signals) and CPU for *Traditional* and *Smart* P-FP (to compute positioning). Fig. 6 shows the energy distribution to compute $p(\mathbf{o}|l)$ versus the number of APs (M) for both approaches. Again, measures are referred to 10^8 positioning runs and have been carried out through the PT tool. Also in this case, only Device 2 has been shown because the other two devices provide similar results.

It is important to point out that, as reported by some papers in the literature, the measurements obtained through PT are not very accurate because they can be biased by unreliable energy models. For example, the work in [55] shows some limits of software-based smartphone energy measurement tools highlighting that the Android WiFi scan implementation has some inefficiencies that are not considered by PT. [55] reports that the duration of the entire scanning operation is composed of three phases: (i) device wake-up (*Head*), (ii) scan itself (*Scan*), and (iii) period between scan completion and device suspension (*Tail*). The overall WiFi scan operation (*Head* + *Scan* + *Tail*) requires about 3 s. This behaviour represents a crucial issue when sequential scans are performed. In detail, if periodic WiFi scans are carried out with a period lower than 3 (s), an overlap between scans may occur, so impacting the numerical values of the energy consumption, but this overlapping is not considered by PT. In consequence, the quantitative evaluation of energy savings that we have measured by PT may be subject to such intrinsic tool inaccuracy, may look different if other techniques are used to measure the consumption and will be assessed more precisely as soon as embedded hardware facilities are available in future smartphone platforms. Anyway, the validity of the reported results is confirmed in terms of the percentage of energy saved by the *Smart* procedure with respect to the *Traditional* one since the scanning period remains constant in both cases and the used tool is the same.

Energy consumption due to WiFi scans is constant to 547 (J) $\simeq$ 0.15 (Wh) in all cases. It depends on neither the number of APs nor the P-FP version because it is related only to the need of keeping the WiFi radio interface active. On the other hand, as expected from the results shown in the previous sections, the energy consumption due to the CPU employment varies with the number of APs and with the used P-FP version. In practice, *Smart* P-FP advantage is exploited only in the CPU. The *Smart* approach allows saving a significant quantity of energy but also changes the power need distribution between WiFi interface and CPU, reducing CPU power requirements drastically. In the *Traditional* case, the CPU consumption, on average, is around 60% of the overall consumed energy. If the *Smart* version is applied, the percentage of energy consumed by the CPU is below 20% of the overall consumed energy. It is worth noticing from Fig. 6 that even for small M values [3, 5, 7, 10], on average, *Smart* P-FP requires less than 55% of energy with respect to *Traditional* P-FP. Energy values averaged over the set of M values [3, 5, 7, 10] are 1408 (J) $\simeq$ 0.39 (Wh) for *Traditional* P-FP and 658 (J) $\simeq$ 0.18 (Wh) for *Smart* P-FP.

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

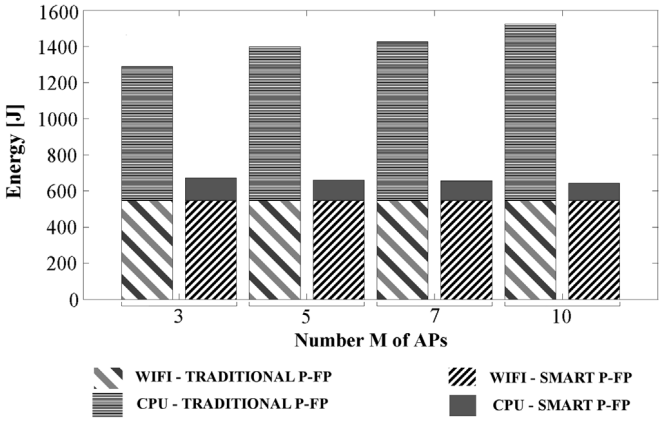


Fig. 6. Energy distribution between WiFi interface and CPU.

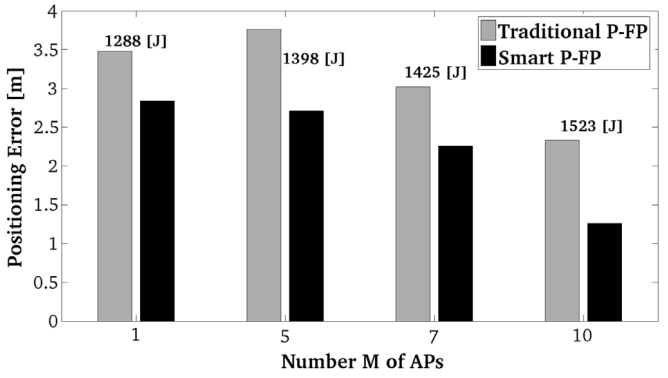


Fig. 7. Trade-off between positioning error (in (m)) and energy for both *Traditional* and *Smart* P-FP.

The results highlight that *Smart* P-FP has a significant impact on the device battery lifetime and, as a consequence, is more suitable to be applied for LBSs and/or CA applications over smartphones.

5.7. Energy distribution between positioning error and WiFi scans

Fig. 7 shows, as a function of the number M of APs, the positioning error for both *Traditional* and *Smart* P-FP keeping fixed the amount of spent energy. Namely, for each pair of histogram bars related to a specific value of M, the overall energy required by the whole positioning process (*i.e.,* WiFi scans and CPU employment) is constant. Its values are reported within Fig. 7. For example, $M = 1$ requires 1288 (J), $M = 3$ needs 1398 (J) and so on. Experimental results show that *Smart* P-FP procedure can obtain a lower positioning error with respect to *Traditional* P-FP when the same amount of APs and energy is used. The motivation of this outcome is the following: if the overall amount of energy and the number M of APs are fixed, the lower CPU employment required by *Smart* P-FP allows keeping active the WiFi radio interface for a longer period of time (see Fig. 6 for reference). As reported at the beginning of Section 3, higher values of T provide a more robust average RSS value and, consequently, lower positioning errors. In practice, if the overall amount of energy and the number M of APs are fixed, *Smart* P-FP can trade energy with a lower positioning error. This fact becomes even more relevant if the number M of APs grows, as shown in Fig. 7.

6. Performance evaluation: The *Smart* $p(\mathbf{o}|l)$ computation to boost Gaussian-based indoor positioning algorithms

One of the strongest points of the proposed *Smart* $p(\mathbf{o}|l)$ computation is represented by its capability to reduce the number of FLOPs necessary to estimate the user position, for all the FP positioning methods that require $p(\mathbf{o}|l)$ based on Gaussian probabilities. To illustrate such improvement, three state of the art algorithms of this type have been considered: (i) Haeberlen et al. [56], (ii) Roos et al. [13] and, (iii) HORUS [28]. For each method, the computational load in terms of number of FLOPs required to estimate the user position has been calculated by using (or not) the *Smart* $p(\mathbf{o}|l)$ computation procedure in Eq. (13). Table 9 summarizes the obtained results and will help show the effectiveness of the proposed procedure. All the calculations necessary to get the formulas reported in such table are provided in the Appendix. M is the number of APs and L the number of RPs, as in Eqs. (5) and (13). V is the number of observation vectors. M' is specified below. Differently from

Table 9
Computational Load of three of the main indoor positioning algorithms with and without the employment of the *Smart* procedure to compute $p(\mathbf{o}|l)$.

Algorithm	Computational load	
	Without the *Smart* procedure	With the *Smart* procedure
HORUS [28]	$4 \cdot \sum_{m=1}^{M'} [88.8m - 2.9] \cdot \lvert\mathbf{L}(m)\rvert$	$4 \cdot \sum_{m=1}^{M'} [8.8m + 6.2] \cdot \lvert\mathbf{L}(m)\rvert$
Haeberlen [56]	$88.8ML + 5.8L - 1$	$8.8ML + 15.9L - 1$
Roos [13]	$[(88.8M - 2.9) + (V + 1) + 2.9] \cdot L$	$[(8.8M + 6.2) + (V + 1) + 2.9] \cdot L$

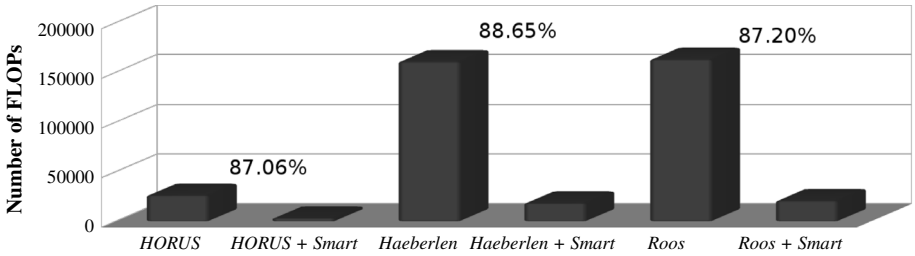


Fig. 8. Comparison of the computational load required by the positioning (online) phase of (i) HORUS [28]. (ii) Haeberlen et al. [56], and (iii) Roos et al. [13] with and without the employment of the *Smart* procedure.

the analysis reported in the previous section, aimed at showing the performance advantage brought by the *Smart* procedure focusing on a specific positioning algorithm, here the number L of RPs necessary to determine the user position must be considered. The three aforementioned algorithms employ different positioning strategies whose number of operations is affected by the number of RPs employed.

For the HORUS algorithm [28], in our comparison, we have assumed that M' (the number of APs necessary to converge to a location estimation) is equal to 6 APs. Furthermore, we have decided that the number of RPs $|\mathbf{L}(m)|$ involved in the positioning process and function of the number of APs, starts from an initial value of 50 with 1 AP and decreases with the number of employed APs *e.g.,* $|\mathbf{L}(1)| = 50$ RPs if only the first cluster is considered, $|\mathbf{L}(2)| = 25$ RPs if the first two clusters are considered, $|\mathbf{L}(3)| = 17$ RPs for the first three and so on. For the other two algorithms [56,13], 200 RPs have been considered. V in [13] is assumed equal to 20. Fig. 8 contains the number of FLOPs computed from Table 9 as explained above for the three considered algorithms both without the *Smart* procedure (just as [28,56,13] do) and with the exploitation of the *Smart* procedure.

The employment of the *Smart* procedure allows saving about 88% of the number of FLOPs that the three considered algorithms require in their original versions (see the numbers over the bars). It is worth remembering that such saving is obtained without any impact on the position accuracy because the *Smart* procedure does not introduce any approximation.

7. Conclusions

This paper proposes a new method to compute the probability of an observation vector at a given Reference Point (RP) based on Gaussian probabilities and a consequent Probabilistic FingerPrint-based (P-FP) method called *Smart* P-FP. It is employed to determine the position of a Mobile Device (MD) in an indoor environment. The key idea is to employ some algebraic factorizations of the equations employed in the *Traditional* P-FP so as to allow computing and storing some quantities directly in the *training* (offline) phase and to avoid many computations during every positioning process (online). The *Smart* P-FP procedure does not apply any approximations and, consequently, does not change the positioning accuracy assured by the *Traditional* P-FP.

An analytical estimation of the computational load required by *Smart* and *Traditional* Probabilistic-FingerPrint (P-FP) has been presented. Such estimation allows deriving important considerations related to the number of FLoating points OPerations (FLOPs) and to the time/energy saved in percentage by using the *Smart* procedure with respect to the *Traditional* one when the number M of used Access Points (APs) is varying. Experiments performed for $M \leq 100$ over three different off-the-shelf Android smartphones, substantially overlap the theoretical values of saved time/energy, so confirming the validity of the proposed approach. The amount of time/energy saved is significant even when few APs are used: $M = 3$ allows *Smart* P-FP saving more than 86% of energy with respect to *Traditional* P-FP. Finally, if M belongs to the range [3, 100], the percentage of saved time/energy is equal or above 90% for all the devices.

Furthermore, the adoption of the *Smart* procedure within three of the most performing positioning algorithms in the state of the art provides about 87%/88% of FLOPs savings with the respect to the original $p(\mathbf{o}|l)$ implementation. This confirms the efficiency of the proposed procedure and opens the door to further practical applications.

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

Appendix

This section provides the details of the comparison among the following positioning schemes: (i) Haeberlen et al. [56], (ii) Roos et al. [13] and, (iii) HORUS [28]. The quantities Ω_O and Ω_S are calculated and reported for each of them. In this Appendix, we reference each of the three investigated algorithms with the term *Original* in case they are considered in their original version while, on the other hand, we use the term *Smart* when the Smart procedure is introduced to compute $P(\mathbf{o}|l)$. For the sake of brevity, only data from Device 1 has been employed for such analysis (see Table 3).

[56] Computational load in the positioning (online) phase

[56] employs a Bayesian localization positioning system that exploits a Markovian approach. The paper estimates the probability that a user is in each RP of the radiomap by using the observation vector $\mathbf{o}$. Consequently, the location is determined by considering the RP which has the highest probability.

To compute such probabilities, the quantities η and $\bar{\pi}_i'$, reported in Eqs. (1) and (2) of [56], must be calculated. Considering these two quantities together allows obtaining the *Original* computational load $\Omega_{O,HB}$ to compute the probability $p(\mathbf{o}|l)$, being L the number of RPs and M the number of APs:

$$\begin{aligned}\Omega_{O,HB} &= [(88.8M - 2.9) + 2.9] \cdot L + (L + 1) + (2 \cdot 2.9)L \\ &= 88.8ML + L - 1 + 4.8L = 88.8ML + 5.8L - 1.\end{aligned} \tag{20}$$

Similarly, if the *Smart* procedure is exploited, the *Smart* + Haeberlen algorithm computational load $\Omega_{S,HB}$ is:

$$\begin{aligned}\Omega_{S,HB} &= [(8.8M + 6.2) + 2.9] \cdot L + (L + 1) + (2 \cdot 2.9)L \\ &= 8.8ML + 9.1L + L - 1 + 4.8L = 8.8ML + 15.9L - 1.\end{aligned} \tag{21}$$

[13] Computational load in the positioning (online) phase

The algorithm proposed in [13] considers a probabilistic positioning system implemented through Gaussian kernel and histogram approach. To achieve a fair comparison with this approach, only the first one has been considered. The positioning algorithm [13] is very similar to the *Traditional* approach reported in this paper. It slightly differs for the formula of the $p(\mathbf{o}|l)$, reported in Eq. (3) of the original paper [13]. Such formula involves a weighted sum over the number of observation vectors V of Gaussian kernels. Consequently, its *Original* computational load $\Omega_{O,R}$ to calculate the probability $p(\mathbf{o}|l)$ for all the considered RPs is:

$$\Omega_{O,R} = [(88.8M - 2.9) + (V + 1) + 2.9] \cdot L. \tag{22}$$

The implementation of the *Smart* procedure allows obtaining the following computational load for the *Smart* + Roos algorithm $\Omega_{S,R}$:

$$\Omega_{S,R} = [(8.8M + 6.2) + (V + 1) + 2.9] \cdot L. \tag{23}$$

HORUS computational load in the positioning (online) phase

Differently from the previous algorithms, the positioning (online) phase of HORUS is composed of four distinct blocks: (i) *Correlation Handler*, (ii) *Discrete-Space Estimator*, (iii) *Small-Scale Compensator*, and (iv) *Continuous-Space Estimator* (see Fig. 6 in [28]). For the computation of $p(\mathbf{o}|l)$ in the HORUS algorithm only the *Discrete-Space Estimator* and the *Small-Scale Compensator* have been considered since the remaining blocks perform operations that are not related to that probability. Both of them employ a product of Gaussians, reported in Eq. (3) of [28]. It is the same formula reported in Eq. (5) of this paper.

HORUS introduces the idea of cluster, which is a set of RPs sharing a common set of APs. The positioning algorithm works as follows. Given the observations $\mathbf{o}$, the APs are sorted in a descending order according to the average RSS. For the first AP, the one with the largest average RSS, the corresponding $p(\mathbf{o}|l)$ is computed for $l \in [1, |\mathbf{L}(1)|]$, where $|\mathbf{L}(1)|$ is the number of RPs belonging to the cluster set of the first AP.[3] If there is a RP l^* with a probability higher than a threshold (set to 0.1), HORUS returns l^* as position estimate. If such RP does not exist, the next AP in the sorted AP list is considered. The process is repeated only for a number of RPs equal to $|\mathbf{L}(2)|$, where $|\mathbf{L}(2)|$ are the RPs belonging both to the clusters 1 and 2. The algorithm stops when a location estimation is reached.

Once HORUS has estimated the position of a user, it can refine its decision by employing the *Small-Scale Compensator*. The system perturbs the observations $\mathbf{o}$ by considering all the possible combinations obtainable by adding and subtracting a small quantity to each AP's RSS within the vector $\mathbf{o}$ (see Section 3.6 of [28]). [28] uses 3 different perturbed observation vectors from the original one $\mathbf{o}$.

[3] $\mathbf{L}(1)$ is the set of RPs for the cluster of AP 1. $|\mathbf{L}(1)|$ represents its cardinality (*i.e.,* the number of RPs belonging to the set $\mathbf{L}(1)$).

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

Consequently, the computational load $\Omega_{O,HS}$ needed by the HORUS positioning system in its *Original* implementation to compute the probability $p(\mathbf{o}|l)$ for all the considered RPs can be written as follows:

$$\Omega_{O,HS} = \sum_{m=1}^{M'} [88.8m - 2.9] \cdot |\mathbf{L}(m)| + 3 \cdot \sum_{m=1}^{M'} [88.8m - 2.9] \cdot |\mathbf{L}(m)|$$
$$= 4 \cdot \sum_{m=1}^{M'} [88.8m - 2.9] \cdot |\mathbf{L}(m)| \quad (24)$$

where the second term of the summation is multiplied by 3 for the *Small-Scale Compensator* contribution. The quantity M', $M' \leq M$ is the number of APs necessary to HORUS to converge to a location estimation.

Similarly to what was already done for the other two algorithms, the computational load for the *Smart* + HORUS algorithm $\Omega_{S,HS}$ is:

$$\Omega_{S,HS} = \sum_{m=1}^{M'} [8.8m - 6.2] \cdot |\mathbf{L}(m)| + 3 \cdot \sum_{m=1}^{M'} [8.8m - 6.2] \cdot |\mathbf{L}(m)|$$
$$= 4 \cdot \sum_{m=1}^{M'} [8.8m + 6.2] \cdot |\mathbf{L}(m)|. \quad (25)$$

References

[1] I. Bisio, F. Lavagetto, M. Marchese, Context-aware smartphone services, in: Pervasive Computing and Communications Design and Deployment: Technologies, Trends and Applications, Hershey, IGI Global, Oxford, 2011.
[2] Y. Wang, X. Yang, Y. Zhao, Y. Liu, L. Cuthbert, Bluetooth positioning using rssi and triangulation methods, in: Consumer Communications and Networking Conference, CCNC, 2013 IEEE, 2013, pp. 837–842. http://dx.doi.org/10.1109/CCNC.2013.6488558.
[3] L. Chen, H. Kuusniemi, Y. Chen, L. Pei, T. Kroger, R. Chen, Information filter with speed detection for indoor bluetooth positioning, in: 2011 International Conference on Localization and GNSS ICL-GNSS, 2011, pp. 47–52. http://dx.doi.org/10.1109/ICL-GNSS.2011.5955258.
[4] Y. Liu, H. Du, Y. Xu, The research and design of the indoor location system based on rfid, in: 2011 Fourth International Symposium on Computational Intelligence and Design, ISCID, vol. 2, 2011, pp. 87–90. http://dx.doi.org/10.1109/ISCID.2011.123.
[5] S. Jeon, J. Park, A rfid reader configuration with an enhanced recognition property for indoor positioning, in: Fifth International Joint Conference on INC, IMS and IDC, 2009. NCM '09. 2009, pp. 166–169. http://dx.doi.org/10.1109/NCM.2009.301.
[6] X. Jinsong, L. Xiaochun, W. Haitao, B. Yujing, B. Yan, W. Chaogang, W. Jing, Design and implementation of channel estimation and equalization of indoor positioning system based on UWB, in: ICICTA '09, 2009, pp. 57–61. http://dx.doi.org/10.1109/ICICTA.2009.730.
[7] M. Kanaan, F. Akgul, B. Alavi, K. Pahlavan, A study of the effects of reference point density on toa-based uwb indoor positioning systems, 2006, pp. 1–5. http://dx.doi.org/10.1109/PIMRC.2006.254047.
[8] D. Zou, F. Liu, Y. Li, T. Han, Study on a joint synchronization algorithm in timedomain for ofdm-uwb based indoor precision positioning system receiver, in: Image and Signal Processing, CISP, 2010 3rd International Congress on, Vol. 9, 2010, pp. 4458–4462. http://dx.doi.org/10.1109/CISP.2010.5647695.
[9] B. Li, J. Salter, A.G. Dempster, C. Rizos, Indoor positioning techniques based on wireless lan, in: LAN, First IEEE International Conference On Wireless Broadband and Ultra Wideband Communications, pp. 13–16.
[10] A.A. Ali, A.S. Omar, Time of arrival estimation for WLAN indoor positioning systems using matrix pencil super resolution algorithm, in: The 2nd Workshop on Positioning.
[11] M. Bocquet, C. Loyez, A. Benlarbi-Delai, Using enhanced-tdoa measurement for indoor positioning, IEEE Microw. Wirel. Compon. Lett. 15 (10) (2005) 612–614. http://dx.doi.org/10.1109/LMWC.2005.855392.
[12] B. Tay, W. Liu, D.H. Zhang, Indoor angle of arrival positioning using biased estimation, in: 7th IEEE International Conference on Industrial Informatics, 2009, INDIN 2009, 2009, pp. 458–463. http://dx.doi.org/10.1109/INDIN.2009.5195847.
[13] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, J. Sievänen, A probabilistic approach to wlan user location estimation, Int. J. Wirel. Inf. Netw. 9 (3) (2002) 155–164. http://dx.doi.org/10.1023/A:1016003126882.
[14] Y. Gu, A. Lo, I. Niemegeers, A survey of indoor positioning systems for wireless personal networks, IEEE Commun. Surv. Tutor. 11 (1) (2009) 13–32. http://dx.doi.org/10.1109/SURV.2009.090103.
[15] Y. Gwon, R. Jain, Error characteristics and calibration-free techniques for wireless lan-based location estimation, in: MobiWac '04, ACM, New York, NY, USA, 2004, pp. 2–9. http://dx.doi.org/10.1145/1023783.1023786.
[16] S. Mazuelas, A. Bahillo, R. Lorenzo, P. Fernandez, F. Lago, E. Garcia, J. Blas, E. Abril, Robust indoor positioning provided by real-time rssi values in unmodified wlan networks, IEEE J. Sel. Top. Sign. Proces. 3 (5) (2009) 821–831. http://dx.doi.org/10.1109/JSTSP.2009.2029191.
[17] S.-H. Fang, T.-N. Lin, K.-C. Lee, A novel algorithm for multipath fingerprinting in indoor wlan environments, IEEE Trans. Wireless Commun. 7 (9) (2008) 3579–3588. http://dx.doi.org/10.1109/TWC.2008.070373.
[18] P. Bahl, V.N. Padmanabhan, Radar: An in-building rf-based user location and tracking system, in: INFOCOM, 2000, pp. 775–784.
[19] T. Kitasuka, T. Nakanishi, A. Fukuda, Wireless lan based indoor positioning system wips and its simulation, vol. 1, 2003, pp. 272–275 http://dx.doi.org/10.1109/PACRIM.2003.1235770.
[20] H. Lim, L.-C. Kung, J.C. Hou, H. Luo, Zero-configuration indoor localization over IEEE 802.11 wireless infrastructure, Wirel. Netw. 16 (2) (2010) 405–420. http://dx.doi.org/10.1007/s11276-008-0140-3.
[21] A. Carroll, G. Heiser, An analysis of power consumption in a smartphone, in: Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 21–35.
[22] I. Bisio, F. Lavagetto, M. Marchese, A. Sciarrone, Gps/hps-and wi-fi fingerprint-based location recognition for check-in applications over smartphones in cloud-based LBSs, IEEE Trans. Multimedia 15 (4) (2013) 858–869. http://dx.doi.org/10.1109/TMM.2013.2239631.
[23] Y. Kim, Y. Chon, H. Cha, Smartphone-based collaborative and autonomous radio fingerprinting, IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 42 (1) (2012) 112–122. http://dx.doi.org/10.1109/TSMCC.2010.2093516.
[24] P. Castro, P. Chiu, T. Kremenek, R. Muntz, A probabilistic room location service for wireless networked environments, in: Ubicomp 2001: Ubiquitous Computing, Springer, Berlin, Heidelberg, 2001, pp. 18–34.
[25] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, B. Schilit, Place lab: device positioning using radio beacons in the wild, in: PERVASIVE'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 116–133. http://dx.doi.org/10.1007/11428572_8.

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001

[26] P. Bolliger, Redpin—adaptive, zero-configuration indoor localization through user collaboration, in: MELT '08, ACM, New York, NY, USA, 2008, pp. 55–60. http://dx.doi.org/10.1145/1410012.1410025.
[27] I. Bisio, F. Lavagetto, M. Marchese, M. Pastorino, A. Randazzo, Trainingless fingerprinting-based indoor positioning algorithms with smartphones using electromagnetic propagation models, in: 2012 IEEE International Conference on Imaging Systems and Techniques, IST, 2012, pp. 190–194. http://dx.doi.org/10.1109/IST.2012.6295537.
[28] M. Youssef, A. Agrawala, The horus wlan location determination system, in: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, MobiSys '05, ACM, New York, NY, USA, 2005, pp. 205–218. http://dx.doi.org/10.1145/1067170.1067193.
[29] L.F.M. de Moraes, B.A.A. Nunes, Calibration-free wlan location system based on dynamic mapping of signal strength, in: MobiWac '06, ACM, New York, NY, USA, 2006, pp. 92–99. http://dx.doi.org/10.1145/1164783.1164799.
[30] M.B. Kjaergaard, A taxonomy for radio location fingerprinting, in: Proceedings of the 3rd International Conference on Location- and Context-Awareness, LoCA'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 139–156.
[31] A. Kushki, K.N. Plataniotis, A.N. Venetsanopoulos, Kernel-based positioning in wireless local area networks, IEEE Trans. Mob. Comput. 6 (6) (2007) 689–705. http://dx.doi.org/10.1109/TMC.2007.1017.
[32] P. Prasithsangaree, P. Krishnamurthy, P. Chrysanthis, On indoor position location with wireless lans, in: The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2002, vol. 2, 2002, pp. 720–724. http://dx.doi.org/10.1109/PIMRC.2002.1047316.
[33] C. Feng, W. Au, S. Valaee, Z. Tan, Received-signal-strength-based indoor positioning using compressive sensing, IEEE Trans. Mob. Comput. 11 (12) (2012) 1983–1993. http://dx.doi.org/10.1109/TMC.2011.216.
[34] M. Brunato, R. Battiti, Statistical learning theory for location fingerprinting in wireless LANs, Comput. Netw. 47 (6) (2005) 825–845. http://dx.doi.org/10.1016/j.comnet.2004.09.004.
[35] N. Swangmuang, P. Krishnamurthy, Location fingerprint analyses toward efficient indoor positioning, of the 2008 Sixth Annual IEEE.
[36] H. Liu, H. Darabi, P. Banerjee, J. Liu, Survey of wireless indoor positioning techniques and systems, IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 37 (6) (2007) 1067–1080. http://dx.doi.org/10.1109/TSMCC.2007.905750.
[37] M. Kjeargaard, G. Treu, C. Linnhoff-Popien, Zone-based RSS reporting for location fingerprinting, in: Pervasive Computing, vol. 4480, Springer, Berlin, Heidelberg, 2007, pp. 316–333. http://dx.doi.org/10.1007/978-3-540-72037-9_19.
[38] N. Brouwers, M. Zuniga, K. Langendoen, Incremental wi-fi scanning for energy-efficient localization, in: 2014 IEEE International Conference on Pervasive Computing and Communications, PerCom, 2014, pp. 156–162. http://dx.doi.org/10.1109/PerCom.2014.6813956.
[39] A. Rice, S. Hay, Measuring mobile phone energy consumption for 802.11 wireless networking, Pervasive Mob. Comput. 6 (6) (2010) 593–606. http://dx.doi.org/10.1016/j.pmcj.2010.07.005. special Issue PerCom 2010.
[40] A.P. Miettinen, J.K. Nurminen, Energy efficiency of mobile clients in cloud computing, in: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 4–11.
[41] P. Bromiley, Products and convolutions of gaussian distributions, Tech. Rep., Medical School, University of Manchester, Manchester, M13 9PT, UK, 2003.
[42] A. Ihler, E.T. Ihler, E. Sudderth, W. Freeman, A. Willsky, Efficient multiscale sampling from products of gaussian mixtures, in: In NIPS 17, MIT Press, 2003, p. 2003.
[43] C. Yang, R. Duraiswami, N. Gumerov, L. Davis, Improved fast gauss transform and efficient kernel density estimation, in: Proceedings. Ninth IEEE International Conference on Computer Vision, 2003, vol. 1 2003, pp. 664–671 http://dx.doi.org/10.1109/ICCV.2003.1238383.
[44] M. Spivak, S.K. Veerapaneni, L. Greengard, The fast generalized gauss transform. URL: http://www.cims.nyu.edu/~shravan/papers/fggt.pdf.
[45] Numerical Recipes in C, The Art of Scientific Computing, second ed., Cambridge University Press, 1992.
[46] Q. Chen, G. Huang, S. Song, Wlan user location estimation based on receiving signal strength indicator, in: 5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009, WiCom '09, 2009, pp. 1–4. http://dx.doi.org/10.1109/WICOM.2009.5305128.
[47] Z. Shen, R. Chen, J. Andrews, R. Heath, B. Evans, Low complexity user selection algorithms for multiuser mimo systems with block diagonalization, IEEE Trans. Signal Process. 54 (9) (2006) 3658–3663. http://dx.doi.org/10.1109/TSP.2006.879269.
[48] A. Laub, M. Heath, C. Paige, R. Ward, Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms, IEEE Trans. Automat. Control 32 (2) (1987) 115–122. http://dx.doi.org/10.1109/TAC.1987.1104549.
[49] A. Scopatz, K.D. Huff (Eds.), Effective Computation in Physics, O'Reilly Media, 2015.
[50] G.P. Perrucci, F. Fitzek, J. Widmer, Survey on energy consumption entities on the smartphone platform, in: Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd, 2011, pp. 1–6. http://dx.doi.org/10.1109/VETECS.2011.5956528.
[51] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, MD, USA, 1996.
[52] M.R. Fernandez, Nodes, sockets, cores and flops, oh, my. URL: http://en.community.dell.com/techcenter/high-performance-computing/w/wiki/2329.
[53] E. Oliver, Diversity in smartphone energy consumption, in: Proceedings of the 2010 ACM Workshop on Wireless of the Students, by the Students, for the Students, S3 '10, ACM, New York, NY, USA, 2010, pp. 25–28. http://dx.doi.org/10.1145/1860039.1860048.
[54] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R.P. Dick, Z.M. Mao, L. Yang, Accurate online power estimation and automatic battery behavior based power model generation for smartphones, in: Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES/ISSS '10, ACM, New York, NY, USA, 2010, pp. 105–114. http://dx.doi.org/10.1145/1878961.1878982.
[55] N. Brouwers, M. Zuniga, K. Langendoen, NEAT: A novel energy analysis toolkit for free-roaming smartphones, in: Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, SenSys '14, ACM, New York, NY, USA, 2014, pp. 16–30. http://dx.doi.org/10.1145/2668332.2668337.
[56] A. Haeberlen, E. Flannery, A.M. Ladd, A. Rudys, D.S. Wallach, L.E. Kavraki, Practical robust localization over large-scale 802.11 wireless networks, in: MobiCom '04, ACM, New York, NY, USA, 2004, pp. 70–84. http://dx.doi.org/10.1145/1023720.1023728.

Please cite this article in press as: I. Bisio, et al., *Smart* probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices, Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.02.001