RSS and Sensor Fusion Algorithms for Indoor Location Systems on Smartphones

Laia Descamps-Vila, A. Perez-Navarro and Jordi Conesa

Abstract Location-based applications require knowing the user position constantly in order to find out and provide information about user's context. They use GPS signals to locate users, but unfortunately GPS location systems do not work in indoor environments. Therefore, there is a need of new methods that calculate the location of users in indoor environments using smartphone sensors. There are studies that propose indoor positioning systems but, as far as we know, they neither run on Android devices, nor can work in real environments. The goal of this chapter is to address that problem by presenting two methods that estimate the user position through a smartphone. The first method is based on euclidean distance and use Received Signal Strength (RSS) from WLAN Acces Points present in buildings. The second method uses sensor fusion to combine raw data of accelerometer and magnetometer inertial sensors. An Android prototype that implements both methods has been created and used to test both methods. The conclusions of the test are that RSS technique works efficiently in smartphones and achieves to estimate the position of users well enough to be used in real applications. On the contrary, the test results show that sensor fusion technique can be discarded due to bias errors and low frequency readings from accelerometers sensor.

L. Descamps-Vila · A. Perez-Navarro (✉) · J. Conesa
IT, Multimedia and Telecommunications Department, Universitat Oberta de Catalunya, Rambla del Poblenou 156, 08018 Barcelona, Spain
e-mail: aperezn@uoc.edu

L. Descamps-Vila
e-mail: ldescamps@uoc.edu

J. Conesa
e-mail: aperezn@uoc.edu

1 Introduction

Location-based applications for smartphones require knowing the location of the user in order to provide information of user's context. That information can be easily gathered in outdoor environments by using Global Positioning System (GPS). However, GPS does not work in indoor environments. Inside a building, the GPS signal is attenuated and scattered by the walls. Therefore, there is a need of alternative location-sensing systems that are able to run on smartphones within indoor environments. Singh et al. 2013, Al Nuaimi and Hesham 2011, Ingram et al. 2004 or Ubisense (Woniak et al. 2013) present different techniques to develop indoor positioning systems.

Among the indoor-location systems, many are wireless driven, i.e. they use of Wireless Local Area Networks (WLAN) present in buildings to position the user: the measurement of Received Signal Strength (RSS) from WLAN Access Points (APs) available in the surrounding space allows to estimate the position of the user.

Fingerprinting is a technique that uses wireless technology to locate users in indoor environments. As Melkonyan 2011 explains, fingerprinting measures, in a preliminary stage called calibration, the RSS from APs of the area at known locations. Such readings are stored as a radio-map database. After that, users at any location can measure same signal features and try to find the statistical match with the radio-map entries and then to find their location on the map.

An algorithm used to estimate position using RSS fingerprinting technique is the nearest reference node that uses the euclidean distance metric (Teuber and Eissfeller 2006). This method compares the RSS values from different APs measured at well-know positions in calibration stage with RSS values measured in an unknown position. The difference between recorded values and current data can be computed as an euclidean distance. It is possible to determine which calibration node is closest to the current position and then convert this distance to a coordinate system.

Gansemer and Pueschel (2010) adapted the basic Euclidean distance algorithm to an environment with changing sets of base stations. However, under specific circumstances still individual heavily incorrect location estimations occur. In addition, the system is not tested on a smartphone.

Ferris et al. (2007) presented Wifi-SLAM project based on a Gaussian Process Latent Variable Model to determine locations using signal strength data. Their library was also used to develop a GIS indoor positioning application (Descamps-Vila et al. 2013). Even though Wifi-SLAM could be integrated in an Android application, calibration data and most of system functionalities were hosted in a web server owned by a company that no longer offers the service. Thus, today the system is no longer available.

As Jung et al. (2012) explain, WiFi-based indoor positioning involve some risks, since in a real situation the infrastructure of WLAN is not controlled. This may lead to different drawbacks: (1) the number of APs emitting is uncontrolled, some APs may be active during the calibration phase but may be inactive when the user estimates the position (see Lee et al. 2013); (2) RSS of each AP is variable and unstable due to reflections, diffractions, multi-path effects, the amount of people or simply the placement of the furniture (Gansemer and Pueschel 2010); (3) user may hold on the smartphone in different position/direction during the calibration and during the measurement stage, which derives in different RSS values (Shala and Rodriguez 2011).

Although there are several methods to estimate indoor location through RSS signals (Lee et al. 2013), (Teramoto et al. 2011), (Hui et al. 2007), as far as we know, there is not any indoor positioning system ready-to-use in smartphones as end-user application and not only in a test and controlled space. To face this problem, the main objective of this work is to design indoor positioning algorithms that can be used as location-sensing systems in smartphones within any indoor space.

The chapter proposes two different algorithms to estimate the user's position in indoor environments. The first one follows a fingerprinting technique, using RSS from WLANs present in the building. This means that the system must overcome problems of WiFi-based indoor positioning and problems of operation performance on smartphones when positioning through WLAN (see Melkonyan 2011).

It is important, also, to preserve accuracy when estimating position. Lee et al. (2013) show that the accuracy of Euclidean distance algorithm can be improved if it is possible to reduce the instability of AP signal strength. Hence, our algorithm has as starting point the algorithm of the Euclidean distance with additional improvements to avoid unstable signals. The second method proposed to estimate user location is the sensor fusion technique, which merge data obtained from different inertial sensors (Woodman 2007), such as accelerometers, gyroscopes and magnetometers, to estimate the movement of the user.

The chapter is structured as follows: Sect. 2 explains the proposed indoor positioning algorithms and an Android prototype that implement them. Section 3 presents the tests done to each system to determine whether they can be used in real situations. Finally, Sect. 4 presents the conclusions and further work.

2 Indoor Algorithm

The indoor location algorithm uses two different techniques: fingerprinting and sensor fusion. The first subsection details how the RSS algorithm estimates user position and the second subsection explains in detail how raw data obtained from smartphone sensors can be combined to estimate user location.

2.1 RSS Algorithm

As explained in Sect. 1, fingerprinting technique is based on the measurement of RSS from different APs. RSS from APs can fluctuate for several reasons such as: a change on the number of people in a room, collisions of radio waves, orientation of mobile receiver, etc. These fluctuations can cause errors on estimating the position. To avoid them, we propose to store only stable data in front of those fluctuations.

As fingerprinting technique is based on two phases, calibration and location, following subsections detail how the fluctuations on both stages can be avoided.

2.1.1 Calibration

On calibration stage there are measurements of RSS at different locations of the building, where there are multiple APs that emit radio signals with different intensities, namely, different levels of RSS.

We call *Node*, N, every location where there is a measurement of RSS during calibration stage. Each node is related to a location in 3D (x, y, z coordinates): $N \in \Re^3$. Every node has associated the measurements of several AP signals: $N_i = [lc_1, ..., lc_k]$ where i is the index of the node N, k is the number of APs visible from the node N and lc_j (level calibration) is the RSS level for each AP (index j) measured in dBm.

In order to store reliable values as calibration data, the algorithm uses two thresholds to discard unstable values. The first one is based on the standard deviation in multiple measurements and the second one limits the number of stored values.

Standard deviation upper threshold

Since values of RSS are very unstable, the measurement of RSS values is repeated n times in each node N_i. Thus, there are n measurements of each AP from each node:

$$N_i(n) = [[{lc_{i,1}}^1, ..., {lc_{i,1}}^n], ..., [{lc_{i,k}}^1, ..., {lc_{i,k}}^n]] \tag{1}$$

where index i represents each Node and index j each AP.

Considering n measurements of RSS values, the standard deviation of RSS values of each AP (index j) by each node (index i) is:

$$\sigma_{i,j} = \sqrt{\frac{1}{n-1}\sum_{r=1}^{n}({lc_{i,j}}^r - \bar{lc}_{i,j})^2} \tag{2}$$

where the mean is defined by:

$$\bar{lc}_{i,j} = \frac{\sum_{r=1}^{n} {lc_{i,j}}^r}{n} \tag{3}$$

The standard deviation is a measure of statistical dispersion. Hence, the algorithm discards APs with high values of $\sigma_{i,j}$ because it means that those APs have many fluctuations. Only the results of APs that are under a threshold are used for the calibration. This threshold has been calculated through the following test.

Table 1 APs standard deviation

	AP1	AP2	AP3	AP4	AP5	AP6
σ (m)	2.134	2.438	2.174	1.569	2.007	2.038

We did 20 measurements of RSS values at a static position. We stored data of six controlled APs. Then, we calculated the standard deviation of each AP. Results are displayed in Table 1.

Results show that all standard deviations are below 3.0 m. Then, the upper threshold to discard unstable measures is 3.0:

$$if\ \sigma_{i,j} \geqslant 3.0 \Rightarrow\ \bar{lc}_{i,j}\ values\ discarded \tag{4}$$

$$if\ \sigma_{i,j} < 3.0 \Rightarrow\ \bar{lc}_{i,j}\ used\ for\ calibration \tag{5}$$

Note that the value of AP signal stored for the calibration map is the mean obtained in Eq. 3.

Limit the number of APs by Node

In some indoor spaces there are many AP signals detected from a single Node, N, sometimes there are over 40 AP signals. Although the algorithm discards the APs measurements that have a standard deviation value over a limit, the calibration map does not need so much data for each Node. On the other hand, as the system runs in a smartphone, the calibration matrix should be of a dimension as low as possible in order to fit the memory requirements of smartphones. Thus, we need to reduce the number of data of calibration and therefore we choose the more stable candidates to calibration data.

To address this issue, standard deviations $\sigma_{i,j}$ of measured APs in a single node N_i are ordered in descendant order. Note that each AP is defined by AP_j and the number of visible APs is defined by k. Thus, the algorithm stores a maximum of k_{max} of AP_j values.

As a result, the $\bar{lc}_{ij}$ with standard deviation $\sigma_{i,j}$ that are in a lower position than k_{max} are discarded because are more unstable than the others. This filtering avoids memory problems and increases performance efficiency, because it limits the size of the calibration matrix.

Calibration matrix

The measurements of RSS that passed both thresholds, define the calibration map, which is modelled as a matrix with the means of RSS levels:

$$Cal\_Matrix = \bar{lc}_{ij} \quad i \in [0, s],\ j \in [0, k] \tag{6}$$

where index i refers to the Node, index j refers to the index of AP, s is the total number of nodes and k the number of APs. Note that the number of nodes in the calibration stage can be defined by the user, but the number of detected APs depends on the WLAN infrastructure of the building and their surroundings.

As explained previously, each Node i has a maximum of k_{max} associated APs. However, each Node could detect different APs. Then, the matrix dimension will not be k_{max}, it will have a variable dimension. In addition, whether a Node, N_i, does not detect an AP_j that is already detected by another Node, the RSS value lc_{ij} will be null, because the Node N_i does not have RSS from the AP.

2.1.2 Location

Once the calibration map is created it is possible to estimate user location. When the user is at an unknown location, the system measures RSS values of APs from the unknown position.

We define the unknown position of the measurement as P, which is associated to a location in x, y, z coordinates: $P \in \Re^3$. Every position P has associated the measures of different AP signals and it is defined as: $P = [lp_1, ..., lp_m]$ where m is the number of APs visible from position P and lp_m (level position) is the RSS value for each AP measured in dBm.

As it is done in calibration stage, the algorithm applies the standard deviation threshold. The system takes n measurements of RSS from a single position P:

$$P(n) = [[lp_1^1, ..., lp_1^n], ..., [lc_m^1, ..., lc_m^n]] \quad (7)$$

$$= \sum_{j=1}^{m} (lp_j{}^1, ..., lc_j{}^n) \quad (8)$$

where index j represent each AP.

To discard unstable measurements, the algorithm apply Eqs. 2–5standard deviation threshold, but only by the index $i = 1$, because the system does measurements from one position P when estimating the location. In this case, instead of a matrix, the results are a list with the means of RSS levels:

$$Measure\_List_j = \bar{lp}_j \quad j \in [0, m] \quad (9)$$

where index j refers to the index of AP and m is the number of APs that passed the standard deviation threshold.

2.1.3 Positioning Through RSS Values

As Teuber and Eissfeller (2006) explains, the RSS signal ratio differences can be expressed in a signal ratio difference vector in which the number of elements represents the number of AP. The ratio difference between nodes N_i and position P can be obtained by euclidean metric:

$$\mathrm{l(P, N_i) = l_i} = \sqrt{\sum_{j=1}^{r} (\bar{lc}_{ij} - \bar{lp}_j)^2} \tag{10}$$

where $\bar{lc}_{ij}$ is the data from the calibration map of Eq. 6, $\bar{lp}_j$ is the location data obtained from Eq. 9 and i is the index of each node used in the calibration stage.

As this algorithm is designed for any indoor environment, it may happen that the detected APs from a node/location may not be always the same. For instance, whether the system makes five measurements in each Node, Node 1 (i=1) may have 5 values for AP_1 (j = 1): $N_1(5) = [{lc_{1,1}}^1, {lc_{1,1}}^2, ..., {lc_{1,1}}^5]$ and four values for AP_2 (j=2): $N_1(4) = [{lc_{1,2}}^1, {lc_{1,2}}^2, ..., {lc_{1,2}}^4]$ because one measurement has not detected the signal of AP2.

Thereby, the algorithm only introduces values in Eq. 10 when there are RSS values from the same APs. This restriction is important to avoid adding non-existent distances. Hence, in Eq. 10, index r defines the coincident APs with signals measured both in calibration and location stages: $AP_r = AP_k \cap AP_m$.

The drawback of taking only the r coincident APs is that we can obtain a smaller distance l_i when there are few APs coincidents than when there are many. Thus, each distance l_i is divided by the number of r values in order to normalize l_i value according to the number of coincident APs. If there is not any coincidence, namely, $AP_r = 0$, the distance cannot be calculated and it is necessary to perform new RSS measures.

Finally, to estimate the position P of a particular location with respect to the coordinate system, Teuber and Eissfeller (2006) suggest using the weighted mean of the coordinates of the closest q calibration Nodes. The selected q Nodes are the ones that have a lower value of l_i. Equations 11 to 13 show how to calculate the coordinates of the position:

$$x = \frac{1}{\sum_{i=1}^{q}(\frac{1}{l_i})} \cdot \sum_{i=1}^{q}(\frac{x_i}{l_i}) \tag{11}$$

$$y = \frac{1}{\sum_{i=1}^{q}(\frac{1}{l_i})} \cdot \sum_{i=1}^{q}(\frac{y_i}{l_i}) \tag{12}$$

$$z = \frac{1}{\sum_{i=1}^{q}(\frac{1}{l_i})} \cdot \sum_{i=1}^{q}(\frac{z_i}{l_i}) \tag{13}$$

where l_i is obtained from Eq. 10 and x_i, y_i, z_i are the coordinates associated to each Node N_i when doing the calibration.

Thus, the position obtained using RSS algorithm is:

$$\mathbf{r}_{RSS} = (x, y, z) \tag{14}$$

At this point we have a first guest of the position. To improve it, in the next step we introduce data from inertial sensors to develop a sensor fusion system.

2.2 Sensor Fusion

Inertial sensors such as accelerometers, gyroscopes and magnetometers can be used to determine the user's movement and location. Thus, these inertial motion sensors may provide data to improve the accuracy of the position obtained with Eq. 14.

Following subsections explain how to estimate the distance traveled and the direction of movement of users using sensor fusion of accelerometer and magnetometer data.

2.2.1 Position Estimation

The accelerometer of a smartphone offers the values of the acceleration (m/s^2) of the device through measurement of the forces applied on the sensor. It provides the values of acceleration in three dimensions of space (x, y, z). Google (R) explains on the documentation that internally the accelerometer sensor used this equation for every single axis:

$$a_i = g_i - \sum_{j=1}^{q} \frac{F_i^j}{m} \tag{15}$$

where i represents each of the coordinate axis, g is the value of the force of gravity, F the force acting on the device, m is the mass of the device and q the number of forces that act on the device.

Equation 15 shows that accelerometer row data includes the gravity force. To measure the real acceleration of the device, the contribution of the force of gravity must be removed from the accelerometer data. The simplest way is to use a high-pass filter to isolate the force of gravity and obtain what is called the linear acceleration, then:

$$a_i' = a_i - g_i \tag{16}$$

where the index i represent each coordinate axis.

Once there is a vector with the acceleration on the three axes, we can estimate the position after some time. It is necessary to apply the equations of cinematics of an object that has a linear movement accelerated. It is shown in Eq. 17:

$$\mathbf{r}_a = \sum_{i=1}^{n} \mathbf{r}_{i-1} + \frac{1}{2}\mathbf{a'}_i \Delta t_{0,i}^2 \tag{17}$$

where $\mathbf{r}_a = (x, y, z)$ is the position after a time Δt, $\mathbf{r_0}$ is the initial position, and a is the acceleration obtained from Eq. 16.

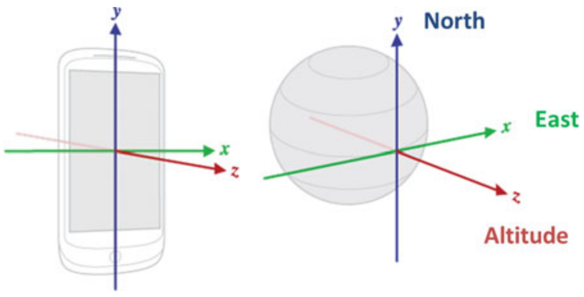


Fig. 1 Smartphone coordinate system (*left*). World coordinate-system defined by Google (*right*)

To obtain the initial position, $\mathbf{r_0}$, we consider that we are still and just in a point where we can obtain the position via GPS or just in a calibration node. Thus, from the starting point and with the accelerometer, we can obtain a new guess of the position: $\mathbf{r_a}$.

2.2.2 Movement Direction

Besides the position estimation, the accelerometer sensor could be useful to determine orientation of movement. The magnetometer measures the strength of the ambient magnetic field in $\mu Tesla$, in x, y, z axes. The combination of both sensors data provides information from the orientation of the device.

Magnetometer and accelerometer measure data in the device coordinates system, but to know the direction the user is moving, it is necessary to know values respect real-world coordinate system. Thereby, raw data obtained from the smartphone must be transformed into another system, which can be used to position the user. This system is defined by Google as *worlds coordinate system*.

The left side of Fig. 1 represents the smartphone's coordinate system. On the right side there is the representation of the coordinate system that Google defined as world's coordinate system. In this coordinate system, x is tangential to the ground at the device's current location and points approximately East; y is tangential to the ground at the device's current location and points toward the geomagnetic North Pole and z points toward the sky and is perpendicular to the ground plane and represents altitude. Both drawings are extracted form Google documentation.

Then, using a rotation matrix defined as an Android internal method, it is possible to transform any vector from the smartphone's coordinate system to the world's coordinate system or vice-versa. The processing of data with the rotation matrix derives into the orientation of the smartphone respect world's coordinate system.

However, the positioning system requires to know the azimuth angle. This is the angle between magnetic north and the device's y axis (Fig. 1). When the device's y axis is aligned with magnetic north this value is 0, if device's y axis is pointing south this value is 180°, when pointing east the value is 90° and when it is pointing west this value is 270°.

To obtain the azimuth angle the system can apply another internal method from Android, called getOrientation. Finally, it is only necessary to transform values obtained from this method from radians into degrees in order to know the angle respect the magnetic north. This lets the user to know the relative direction that is oriented the device respect to Earth frame of reference, because the Earth's magnetic north is known. It works in a similar way to traditional compasses.

2.3 Prototyping

We implemented the algorithms and methods described in this section on an Android prototype. The objective is to test whether all of them can be used in an Android smartphone and whether it is possible to estimate the position in an indoor environment.

The main functionalities of the prototype are: to display the available floor plan of buildings; to calibrate the floor plan with RSS values (see Sect. 2.1); to measure the position of the Android device using the accelerometer (see Sect. 2.2.1); to show the direction of movement of the user respect north magnetic pole like a compass (see Sect. 2.2.2); to show the estimated user position over the floor map.

3 Testing

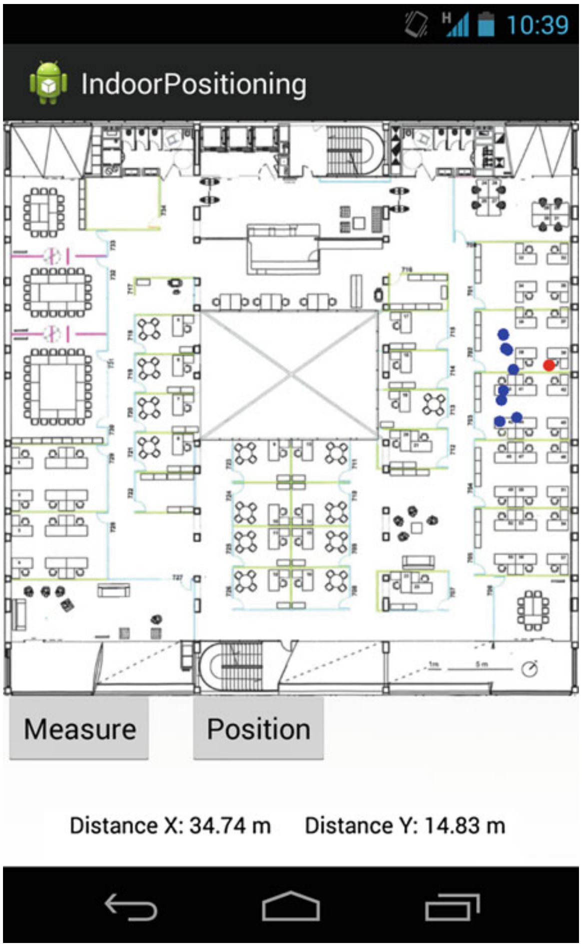
The smartphone used to perform the tests is a Galaxy Nexus with Android 4.3. The first subsection presents tests performed on the prototype with the RSS algorithm and the second subsection shows the tests performed using sensor fusion.

3.1 RSS Tests

This subsection shows the tests performed with the prototype of the RSS positioning algorithm described in Sect. 2.1 within the Android device. We did positioning tests in two different buildings in order to analyze the behavior of the system in two different environments. One place is a flat (Building A) and the other is an office building (Building B). In these tests we used only one floor for each building, what means that the coordinate z is always equal to 0. In addition we dissociated the Wi-Fi connection of the smartphone.

The metrics for the tests are the same in each building. For the calibration we measured 40 nodes: $s = 40$. In each node we repeated the RSS measurement 5 times: $n = 5$. The limit of AP signal values stored by the matrix is 15 for node, $k_{max} = 15$. The standard deviation threshold is 3.0, as defined in Sect. 2.1.1.

We measured 10 different positions in each building. In order to have statistical values, we estimated the position of each one 10 times (therefore, we had 100



Distance X: 34.74 m Distance Y: 14.83 m

Fig. 2 Building B—Position 4

measurements for building). For each estimation of the position, the algorithm took 5 measurements, $n = 5$ and it used the fourth nearest nodes to calculate the x, y coordinates, $q = 4$.

3.1.1 Results

Table 2 presents the results of building A and Table 3 presents the results of Building B. Tables show the mean of the 10 measurements for each position, the error, the standard deviation and the accuracy respect to the theoretical x, y position.

Table 2 Positioning tests in building A

Position	Mean ± error (m)		σ (m)		Accuracy (m)	
	x_p	y_p	σ_x	σ_y	x	y
1	6.94 ± 0.09	9.31 ± 0.16	0.30	0.52	−0.29	0.38
2	3.40 ± 0.01	9.97 ± 0.01	0.01	0.01	−0.53	−0.96
3	4.79 ± 0.02	7.70 ± 0.01	0.06	0.01	−0.23	−0.25
4	5.53 ± 0.22	3.99 ± 0.10	0.68	0.33	1.25	0.42
5	8.65 ± 0.35	3.97 ± 0.19	1.11	0.61	−0.78	−0.65
6	6.42 ± 0.14	8.42 ± 0.09	0.44	0.29	−1.27	1.06
7	5.33 ± 0.18	5.11 ± 0.17	0.57	0.55	−0.24	1.18
8	6.92 ± 0.04	11.07 ± 0.01	0.13	0.02	0.42	−0.26
9	8.03 ± 0.18	1.83 ± 0.09	0.59	0.29	−1.23	0.66
10	6.89 ± 0.08	3.25 ± 0.08	0.26	0.24	0.32	1.30

Table 3 Positioning test building B

Position	Mean ± error (m)		σ (m)		Accuracy (m)	
	x_p	y_p	σ_x	σ_y	x	y
1	35.04 ± 0.31	7.01 ± 0.08	0.99	0.26	−3.02	−0.43
2	32.81 ± 0.32	10.25 ± 0.58	0.92	1.64	0.64	−1.08
3	30.59 ± 0.32	12.48 ± 0.42	1.00	1.31	3.59	2.76
4	34.92 ± 0.12	17.34 ± 0.76	0.37	2.41	−3.08	5.01
5	34.50 ± 0.32	26.76 ± 0.90	1.01	2.84	−3.33	−1.18
6	36.05 ± 0.57	32.90 ± 0.52	1.82	1.63	−0.84	−2.77
7	7.15 ± 1.91	22.63 ± 1.01	6.05	3.19	3.43	−2.26
8	5.85 ± 0.61	15.52 ± 0.59	1.92	1.89	1.79	1.80
9	28.65 ± 0.26	26.36 ± 0.82	0.82	2.60	−0.18	−4.20
10	23.75 ± 0.08	23.58 ± 0.32	0.25	1.01	3.42	−1.36

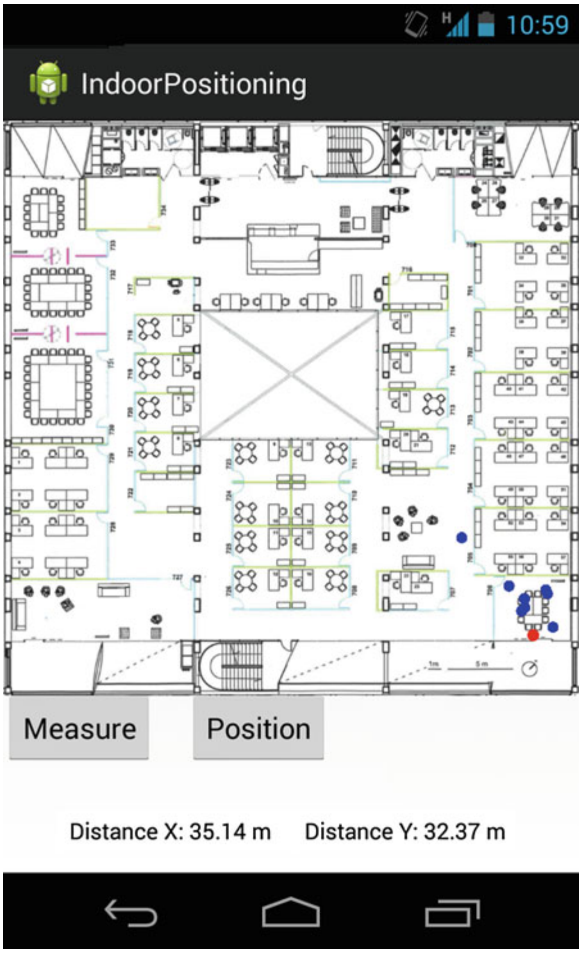
Figures 2 and 3 show two screen-shots of the Android prototype with the tests on two different positions of a Building. Blue dots are the measured positions (results of Tables 2 and 3) and red dot is the theoretical position.

3.1.2 Analysis

The algorithm designed to estimate the position through RSS values provide reasonably good results. In building A results show an accuracy below 1.5 m, however, in building B accuracy is around 2–3 m, except in two positions where the accuracy is around 4–5 m.

The standard deviation of all results, except position 7 of building B, are within 1–2 m of deviation. Thus, the scattering of results is low, as intended in the design of the algorithm. Standard deviation is also better in building A than in building B.

Results show the position measurement is more precise and more accurate in building A. This may be due to two reasons: (1) the number of nodes calibrated is the same in the two buildings, $s = 40$, however building A is smaller than building



Distance X: 35.14 m Distance Y: 32.37 m

Fig. 3 Building B—Position 6

B and therefore there are more nodes calibrated by square meter which improves the calibration; (2) we have seen that in building A there are less number of APs detected, the ones detected are stronger and are always the same. Hence, as stated on Introduction section, the more stable and controlled are the APs, the better is the position estimation.

In addition, note that the theoretical point is not within the margins of error of the measured point. Therefore there are sources of error that are not controlled.

Table 4 Orientation measurements at rest

Mean of angle respect north (degrees)	σ
65.4795	0.8391

3.2 Sensors Tests

Once shown that user position through RSS algorithm can be estimated, this section uses the prototype to test the sensor fusion system proposed in Sect. 2.2 to get the orientation and position of the device.

3.2.1 Movement Direction

This subsection shows that we achieved also to obtain the direction of the device while user is walking. Figure 4 presents the system that we implemented following explanations of Sect. 2.2.2. It is a screenshot of the prototype.

The system indicates the direction a person is moving. The black line inside the circle indicates the Earth's magnetic north and the number shows the orientation to the north. In this case, the head of the smartphone is oriented at 65.5°. In addition, Fig. 4 displays raw data of accelerometer and magnetometer, which are the source data necessary to find this angle.

In order to know the reliability of the obtained values, we measured 2,000 samples of orientation values at a static position. Results are displayed on Table 4. They show the dispersion on data values is low since standard deviation is small.

3.2.2 Position

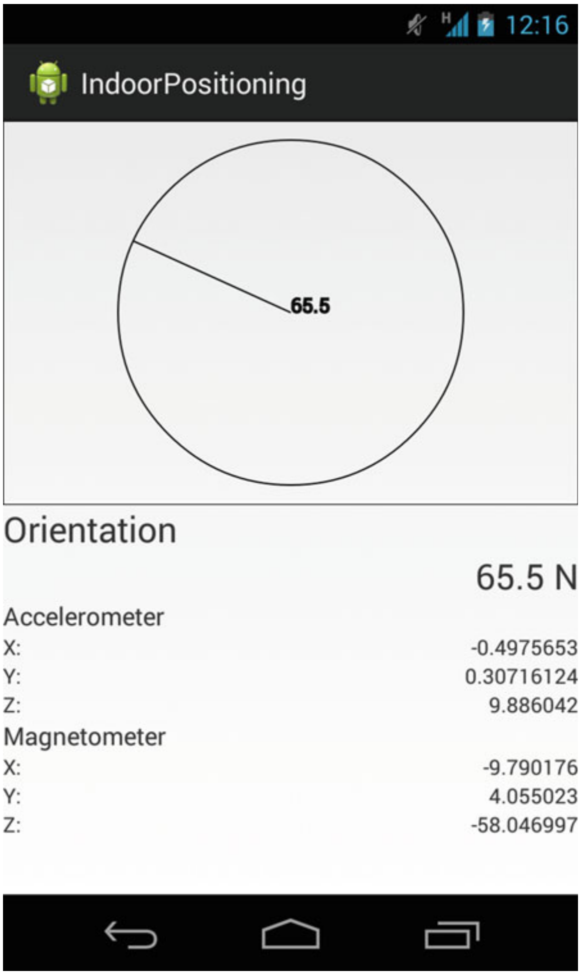
This subsection shows the tests performed with the prototype to estimate the distance traveled by the user, using the equations described in Sect. 2.2.1.

The prototype measured the acceleration while a user is holding the device horizontally and walking in only one direction. Hence, Eq. 17 is used only in one axis, in this case y axis of the smartphone coordinate system (refer to Fig. 1).

Figure 5 presents a graphic with the acceleration raw data measured in the three axes. Note that the raw acceleration from the sensor is the one defined in Eq. 15. The z axis shows the acceleration produced by the force of gravity over the device. y axis shows the acceleration values on the direction of movement because we have shown this direction as the moving axis, and x axis is perpendicular to the direction of movement. The variations shown on the graphic are due to step movement of the user. Hence, it is clear that this data is influenced by the force of gravity.

After isolating the force of gravity and applying Eq. 17 we measured 5 times the same distance.

The distance traveled by the user that is holding the device is 6 m. Results of the tests gave a distance equal to $d_a = 10.801 \pm 0.119\ m$. Results are far away from the



Orientation

65.5 N

Accelerometer

X:	-0.4975653
Y:	0.30716124
Z:	9.886042

Magnetometer

X:	-9.790176
Y:	4.055023
Z:	-58.046997

Fig. 4 Android prototype that shows device movement direction

real distance. These bad results can be caused by a low rate of update readings and bias errors.

If the frequency of data collection is low, the system loses information. For instance, imagine the last measured acceleration is low and then there is a significant increase. In case the sensor takes time to read these data, the information is lost and increases the error.

To test this hypothesis, we took measurements of 2,000 samples of accelerometer values in a static position. Table 5 shows the rate of readings of accelerometer values. The first column displays the mean of the frequency, which is about 60 ms, the second column shows the standard deviation, the third column the maximum value of update

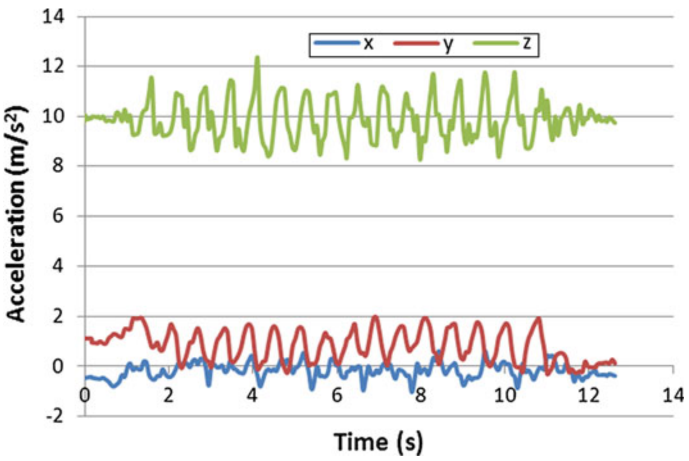


Fig. 5 Accelerometer measurements while user's walking

Table 5 Accelerometer frequency measurements

Mean (ms)	σ (ms)	Max (ms)	Min (ms)
63.73	13.65	133.27	3.81

Table 6 Accelerometer measurements at rest

Axis	Mean (m/s^2)	σ (m/s^2)	Accuracy (m/s^2)
x	−0.497	0.037	−0.497
y	0.119	0.038	0.118
z	9.919	0.049	0.109

reading and the fourth column the minimum. These results show that accelerometer sensors have a low rate of update readings and it is very unstable.

Additionally, the measured distance varies greatly depending on the time it takes to make the movement. We noticed that when the device is at rest, the distance increases constantly, instead of remaining constant. This may be due to the accelerometer have a initial bias error that corresponds to the offset of its output signal from the true value (see report from (Woodman 2007)). A constant bias error of accelerometer causes an error in position which grows quadratically with time.

We measured the bias of the accelerometer from the 2,000 samples measurements defined previously. Table 6 shows the results.

The average of acceleration values is different from the theoretical values (0, 0, 9.8). This is what makes the distance to increase constantly increases even though the device is at rest, because there are a constant bias of the accelerometer values that are integrated over time, the error is cumulative.

In order to avoid this issue, it is therefore necessary to subtract the bias error (mean values of Table 6) during the measurements. However, in a real application, it is not possible to calibrate each smartphone constantly to reduce the bias, because we have seen that the bias is not always the same for the same accelerometer.

After doing several sensor tests, we have seen it is possible to know the direction of movement, however, the accelerometer sensors have a low update reading results and bias error which derive into very bad distance measurement results. Hence, at the moment we avoid using accelerometer to estimate user position.

4 Conclusions

In this chapter two indoor positioning algorithms that can run entirely in a smartphone device have been presented. One of them uses RSS values of WLANs present in a building to estimate the user position. The algorithm is based on the euclidean distance method and it is designed to be used in a smartphone, taking into account the performance limitations of these devices. The other algorithm uses the sensor fusion technique, which combines inertial sensors raw data to detect the direction of movement and distance travelled by the user.

Both algorithms have been implemented in an Android prototype in order to demonstrate that they can work efficiently in such devices and to test their effectiveness. With the prototype, the RSS algorithm has been tested in two different buildings, giving good results. In fact, results show that most of the position measurements have accuracies around 1–3 m and standard deviations of 1–2 m. Comparing results of two buildings, we have seen that, the more nodes are measured during calibration and the stronger and stable the APs of the building, the better the results.

Sensor fusion technique allowed to find out what direction the user is moving, but tests show very bad results when calculating the user position with the accelerometer sensors. The problem seems to be that the accelerometer sensor has a big bias error and a low and unstable frequency of reading results, which derives on big errors when calculating the distance traveled by a user.

Therefore, the main contributions of this chapter are two fold. On the one hand, it presents a couple of algorithms that use smartphone sensors to locate users in indoor environments and demonstrate that they can be implemented and work efficiently in smartphones. On the other hand, it test both algorithms and identify some problems that worsen the quality of location estimation when using data from inertial sensors.

As future work, we plan to do tests on different floors of the two tested buildings, taking into account the z axis in order to analyze the accuracy and precision of the system over different floors. We also would like to establish a calibration standard that works as a guide for the users and to study different methods to address the errors in sensors such as the accelerometer.

Acknowledgments This article has been developed with the support of the Internet Interdisciplinary Institute of the Universitat Oberta de Catalunya.

References

Al Nuaimi K, Hesham K (2011) A survey of indoor positioning systems and algorithms. In: 2011 international conference on innovations in information technology, pp 185–190. doi:10.1109/INNOVATIONS.2011.5893813

Descamps-Vila L, Perez-Navarro A, Conesa J (2013) Integración de un sistema de posicionamiento indoor en aplicaciones SIG para dispositivo móvil. In: VII Jornadas SIG Libre de Girona, 1. http://www.sigte.udg.edu/jornadassiglibre/uploads/articulos_13/a29.pdf

Ferris B, Fox D, Lawrence N (2007) WiFi-SLAM using gaussian process latent variable models. In: International joint conference on artificial intelligence IJCAI, pp 2480–2485. http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-399.pdf

Gansemer S, Pueschel S (2010) Improved RSSI-based euclidean distance positioning algorithm for large and dynamic WLAN environments. Int J Comput 9(1):37–44. http://archive.nbuv.gov.ua/portal/natural/computing/2010_1/PDF/10SGADWE.pdf

Hui L, Darabi H, Banerjee P, Jing L (2007) Survey of wireless indoor positioning techniques and systems. IEEE Trans Syst Man Cybern Part C 37(6):1067–1080

Ingram SJ, Harmer D, Quinlan M (2004) Ultrawideband indoor positioning systems and their use in emergencies. In: Position location and navigation symposium, PLANS 2004, pp 706–715. doi:10.1109/PLANS.2004.1309063

Jung WR, Bell S, Petrenko A, Sizo A (2012) Potential risks of WiFi-based indoor positioning and progress on improving localization functionality. In: International workshop on indoor spatial awareness. http://dl.acm.org/citation.cfm?id=2442621

Lee JY, Yoon CH, Hyunjae P, So J (2013) Analysis of location estimation algorithms for WiFi fingerprint-based indoor localization. In: The 2nd international conference on software technology, vol 19, pp 89–92. http://onlinepresent.org/proceedings/vol19_2013/23.pdf

Melkonyan A (2011) Integrity monitoring and thresholding-based WLAN indoor positioning algorithm for mobile devices. In: The 9th international system of systems engineering conference, pp 191–196. doi:10.1109/SYSOSE.2011.5966596. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5966596. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5966596

Shala U, Rodriguez A (2011) Indoor positioning using sensor-fusion in android devices. Ph.D. thesis, school of health and society department computer science embedded systems. http://hkr.diva-portal.org/smash/get/diva2:475619/FULLTEXT02.pdf

Singh R, Sharma S, Mohan R (2013) A study and analysis of wireless based localization and motion processing systems for healthcare applications. Int J Emerg Technol Adv Eng 3(7):406–414. http://www.ijetae.com/files/Volume3Issue7/IJETAE_0713_68.pdf

Teramoto Y, Sato A, Asahara A, Tomita H (2011) Indoor positioning based on radio signal strength distribution modeling using mirror image method. In: Workshop on indoor spatial awareness, Nov, pp 15–22. http://dl.acm.org/citation.cfm?id=2077361

Teuber A, Eissfeller B (2006) WLAN indoor positioning based on Euclidean distances and fuzzy logic. In: Proceedings of the 3rd workshop on positioning, navigation and communication, pp 159–168. http://www.wpnc.net/fileadmin/WPNC06/Proceedings/31_WLAN_Indoor_Positioning_Based_on_Euclidean_Distances_and_Fuzzy_Logic.pdf

Woniak M, Odziemczyk W, Nagrski K (2013) Investigation of practical and theoretical accuracy of wireless indoor positioning system ubisense. Rep Geodesy Geoinformatics 95(0). http://www.reports.gik.pw.edu.pl/index.php/reports/article/view/233

Woodman O (2007) An introduction to inertial navigation. Technical Report UCAM-CL-TR-696. University of Cambridge, Computer Laboratory. http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf