Multi Sensor System for Pedestrian Tracking and Activity Recognition in Indoor Environments

Juan Jose Marron, Miguel A. Labrador
and Adrian Menendez-Valle
Department of Computer Science and Engineering
University of South Florida Tampa, Florida, USA
Email: jmarronm@mail.usf.edu, labrador@cse.usf.edu
adrian16@mail.usf.edu

Daniel Fernandez-Lanvin
and Martin Gonzalez-Rodriguez
Department of Computer Science
University of Oviedo, Oviedo, Spain
Email: dflanvin@uniovi.es
martin@uniovi.es

***Abstract*—The widespread use of mobile devices and the rise of Global Navigation Satellite Systems (GNSS) have allowed mobile tracking applications to become very popular and valuable in outdoor environments. However, tracking pedestrians in indoor environments with Global Positioning System (GPS)-based schemes is still very challenging. Along with indoor tracking, the ability to recognize pedestrian behavior and activities can lead to considerable growth in location-based applications including pervasive healthcare, leisure and guide services (such as, hospitals, museums, airports, etc.), and emergency services, among the most important ones. This paper presents a system for pedestrian tracking and activity recognition in indoor environments using exclusively common off-the-shelf sensors embedded in smartphones (accelerometer, gyroscope, magnetometer and barometer). The proposed system combines the knowledge found in biomechanical patterns of the human body while accomplishing basic activities, such as walking or climbing stairs up and down, along with identifiable signatures that certain indoor locations (such as turns or elevators) introduce on sensing data. The system was implemented and tested on Android-based mobile phones. The system detects and counts steps with an accuracy of 97% and 96.67% in flat floor and stairs, respectively; detects user changes of direction and altitude with 98.88% and 96.66% accuracy, respectively; and recognizes the proposed human activities with a 95% accuracy. All modules combined lead to a total tracking accuracy of 91.06% in common human motion indoor displacements.**

***Keywords*—Smartphones, Sensor Fusion, Pervasive Computing, Inertial Navigation, Ubiquitous Localization**

I. Introduction and Motivation

Mobile computing devices such as smartphones, tablets and smartwatches are nowadays overtaking the popularity of conventional desktop computers. The computing paradigm has evolved in recent years reducing the prices of these devices while increasing the number of features, processing power and mobility capabilities. As a result, location-aware services and applications have spurred particularly in the wellness [26] and in the entertainment sector [19]. This rapid growth in people-centric mobile computing applications has led to improvements in localization technologies, not only in terms of localization accuracy, but also across multiple and specific dimensions such as power consumption, energy efficiency, response time, and ubiquity.

In outdoor environments localization is successfully solved by traditional Global Navigation Satellite Systems (GNSSs), such as the Global Positioning System (GPS), cell tower localization and Wi-Fi. However, these technologies cannot track the user's position accurately in indoor environments where humans spend most of their time (offices, home, schools, universities, malls, hospitals, etc.). A location service capable of providing accurate positioning in indoor environments could promote the interest in a whole range of new mobile applications in different domains, such as healthcare, transportation, tourism, and many others. The list below includes some of the most important services that could be provided in indoors environments, in this case customized to hospitals and healthcare services.

- Real-time tracking: patients could be tracked continuously for their safety and security inside hospitals.
- Safety: location systems could provide rescue services with an accurate and immediate knowledge of the user's position inside a hospital building in case of emergency.
- Resource-efficiency: hospital could utilize the information of where the patients are to optimize resources such as air conditioning, heating, or lighting.
- Security: location-awareness could permit automatic locking of sensitive resources depending on the owner presence and could trigger alarms if patients cross certain boundaries.
- Social networking: patients and medical professionals could easily and efficiently find each other.
- Automatic resource routing: follow-me applications could allow patients to be routed to achieve their goals.
- Navigation: patients and visitors could be easily guided to navigate to areas or rooms of interest.
- Announcements: messages or alerts could be sent to patients according to their location in the hospital.

Although several systems and solutions have been proposed for indoor localization and tracking, they have not been very successful so far because of the following common challenges:

- Additional hardware. Most proposed systems require some form of supplementary hardware and additional infrastructure that makes them impractical and costly for most applications. Specific high quality sensors are

usually mounted in bare functional locations making the system impractical and uncomfortable for the user's common use.

- Unreliable sensor data. Several obstacles in indoor environments, such as machines, walls, corridors, open areas, metals, etc., introduce random noise in the sensor measurements.
- Cumulative errors. Low cost sensors embedded in mobile devices are normally low quality devices that would lead to cumulative errors when estimation of new positions are based on previous Pedestrian Dead Reckoning calculations.
- Data fusion. The multisensory approach of the proposed system introduces the challenge of combining sensing data collected from sensors of different nature to extract reliable signatures and patterns.
- High level of location accuracy required. Because the indoor context varies at fine spatial granularity, most indoor-based applications require a high level of accuracy from the location system.
- Energy consumption. The continuous use of sensors and processing needs of the location system consume extra energy from already energy-constrained devices like smartphones.
- Processing power. Although hardware resources in mobile devices are in continuous evolution, processing power is still limited. Algorithms and data processing techniques implemented in these devices should prevent a high consumption of resources.
- Mobile device position. The use of a mobile device as data collector introduces the challenge of multiple possible positions and orientations of the device, such as calling, messaging, swinging if the users hold the device in hand while walking. These different actions affect the tracking and activity recognition algorithms.
- Evaluation tools. The evaluation of positioning systems is a tedious job consisting of the repetition of experiments and comparisons of the results with real scenarios. In most cases, experiments are performed manually and individual evaluation is necessary to compute global results.

In this paper, we propose a smartphone-based system for pedestrian tracking in indoor environments that addresses most of these problems. First, the proposed system simultaneously harnesses sensor-based dead-reckoning and environment sensing for localization and does not require of previous calibration nor installation of external infrastructure/additional hardware. It applies a set of data processing techniques to the noisy raw sensor data received from the array of inertial sensors embedded in the smartphone (triaxial orthogonal accelerometer, gyroscope, magnetic field detector and barometric pressure sensor) to produce acceptable location accuracy results. The system implementation is based on two sequential components: a decision tree for activity recognition, which recognizes human indoor activities, such as turns (T), stationary times (Sy), use of elevators (E), walking (W) and stairs (St), and a set of algorithms that provide additional information about the activity, such as motion distance or direction. The algorithms avoid machine learning techniques, which require training tasks, rely on statistical analysis, and demand more processing power. Instead, they are based on filters, peak detection, and thresholds, and use simple internal calibration routines to detect and count stairs and steps, detect turns and altitude changes or recognize motion direction and sensing patterns in indoor environments. In addition, the system uses track-splitting and landmarking strategies to reduce the accumulated error usually inherent in every inertial sensor-based system. The system shows that adequate points of interests (POI) exist in indoor environments that they can be used to make dead-reckoning practical and reasonably accurate. Finally we also introduce and use two tools to evalute the system. The first one and used in this paper consist of an Android application in the client side, which is in charge of collecting the sensor data from the mobile device, and a server application working as a testbed that simplifies the system's implementation and the evaluation process. The second one is a solution for real-time tracking of pedestrians in indoor environments [27].

The remainder of the paper is organized as follows. Section II describes previous research work done in this area. Section III describes the groundwork of this research where the key concepts of the system design are explained including the modules and algorithms that are part of the system. Section IV details the system implementation. Section V includes the evaluation methodology and discusses the performance the individual algorithms and the system as a whole. Finally, Section VI concludes the paper and sets forth directions for future research.

II. Related Works

There is a large body of literature on indoor positioning systems. A comprehensive coverage is provided in surveys related to inertial systems [13] and wireless positioning systems [18] [12]. The set of solutions available can be classified as: lateration and angulation systems, proximity systems, radio fingerprint systems, dead-reckoning systems, and hybrid systems.

Among the current technologies, the lateration and angulation methods are probably the most complex to deploy and expensive to maintain in terms of infrastructure. Similarly to the GPS principle for position estimation, the system functionality is based on the computation of distances between the mobile unit and an array of base stations installed in the building. Careful choice of the beacon signal can contribute to mitigate the difficulties of radio signal propagation that arise due to indoor obstacles. The most common signal types are ultrasonic systems [11], radio frequency-based systems [20], and ultra wideband (UWB) radio systems [14]. As an example, in [31], Zhao et al. propose AUITS, an autonomous ultrasonic indoor tracking system for locating and tracking mobile objects inside a building. The results of their work show that the coverage area to estimate the location of one device can reach up to $65m^2$, obtaining a positioning error of less than $15cm$ with 90% probability or more. However, as most conventional ultrasonic location systems, it poses some challenges such as manual calibration of the transmitters, high installation cost, antenna mismatch, external interferences from other systems, and low power emission.

Proximity systems are another common alternative for indoor positioning. Mobile stations include detectors which recognize signals transmitted by a proximity system. Since the location of the transmitters is known, the true location of the mobile device is easily obtained. Due to the short range of the signals, these systems provide only the nearest room or building area (a coarse location) rather than a coordinate location. Examples include Bluetooth stations, Radio-Frequency Identification (RFID) systems and Near Field Communication (NFC) infrastructures. Bluetooth is a wireless communication technology for the exchange of data in a short range. The position accuracy is proportional to the number of cells used and the precision of the receivers [4]. In the case of RFID technology, the position accuracy also depends on the amount of tags used and the type of these tags, which can be either active or passive. Proposed RFID-based indoor navigation solutions require an extensive usage of tags to get a reliable position and they are generally based on active RFID tags. Actives tags increase the transmitting distance compared with passive tags since they include batteries to increase the transmitting power. The main disadvantage of the solutions based on active RFID tags is the high cost of the transmitters. Furthermore, studies suggest that they do not provide an efficient tracking system [22]. Similarly, NFC solutions [23] require a large number of readers to obtain reliable ubiquitous coverage.

Radio fingerprinting approaches have been the successful indoor systems to date. In these systems, a radio map of various signal properties such as received signal strength is previously collected and compared to the current measurements. The closest match is searched and identified as the estimated position. Wi-Fi is the most common radio fingerprinting choice due to its ubiquity [17] [5] [24]. The Wi-Fi based system typically reports accuracies of a few meters. However, the time required to install, configure and maintain these indoor systems have so far limited their general deployment.

Other popular set of systems are independent navigation systems based on Pedestrian Dead Reckoning (PDR) techniques. Instead of providing a coordinate location, a common method for human tracking is to calculate the current position based on the last estimated position, the speed of the item, the route, and the elapsed time between the current and the previous position. Recently, PDR systems use Micro Electro-Mechanical Sensors (MEMSs) and inertial sensors embedded in cell phones, so very basic physical infrastructure is needed. Furthermore, MEMs based systems usually offer an additional degree of privacy since the user can choose to either share their location information with any third party or not. The most important drawback of PDR-based navigation systems is the need to correct the noise associated with the sensors when the estimation of the new position is based on a previous PDR calculation. For example, the inertial sensors distance traveled can be calculated from the acceleration signal by double integration with respect to time; however, due to the low accuracy of the accelerometer, the presence of noise and the component of acceleration caused by gravity, error accumulates rapidly with time [10]. An interesting approach shown by Constandache et al. in [9] takes advantage of a smartphone's digital compass and accelerometer to track pedestrians. The system is designed and tested for outdoor environments where map information is available. It compares the estimated path with the true map information without requiring any external extra device. Alzantot et al. [2] show how a step counter for tracking pedestrians can be created using exclusively the inertial sensors built in a cell phone. They use dead reckoning navigation techniques combined with lightweight finite state machines to obtain an acceptable accurate level. Additionally, recent developments in PDR systems have incorporated urban sensing and activity recognition. For instance, several inertial sensors worn simultaneously on different parts of the body can detect when a user is walking, turning, or climbing up stairs [28]. Similarly, microphones and magnetometers can be used to detect ambient sounds and magnetic fluctuations [3] [7]. While these signatures have been primarily used for various forms of context awareness, they can contribute to localization purposes as well. These signatures can be treated as landmarks and are useful for indoor dead-reckoning systems when combined with sensor information in order to recognize indoor points of interest and users movements.

All these techniques for indoor positioning systems are not independent and several hybrid systems are also found in the literature. Lateration and angulation systems are often combined with other indoor positioning techniques to improve the global performance. For instance, in [16], Jin et al. propose the use of the digital compass and the accelerometer in a smartphone to track user location in indoor environments. The commercial system, called SparseTrack, uses additional ultrasonic sensors sparsely distributed in the area to correct the possible error provided by the smartphone. In [30], foot-mounted inertial sensors are used to build RF radio maps for pedestrian tracking. The scheme uses a particle filter and the known radio-frequency map information to track the user and to improve the positioning estimated by the basic INS system. In [25], Tom et al. present an almost self-deployable solution based on RFID tags inertial Micro Electro Mechanical Sensors. Capelle et al. [6] designed a GNSS-based multisensory system based on the fusion of three different technologies: high sensitivity GNSS (GPS and the future Galileo), MEMS-based Pedestrian Navigation System and WI-FI. Proposed in [29], Woodman et al. implement a pedestrian localization system for indoor environments using a foot-mounted inertial unit and a localization algorithm using Wi-Fi signal strength to reduce initial complexity. All these hybrid designs partially solve the indoor localization problem. However, all of them require offline training in order to build a radio map or the installation of additional infrastructure. Requiring infrastructure beyond the common mobile phone can make a solution impractical for several scenarios and it will undermine the use of these solutions in real environments.

Comparing this paper with the current literature, the presented work focuses on the integration of raw data and context information collected by sensors embedded in mobile phones only. The cumulative errors, as a consequence of using low quality inertial sensor and PDR techniques, are reduced by splitting and resetting the complete trajectory in small independent traces. Altogether, the proposed method provides an acceptable accuracy at a low cost.

III. System Design

The proposed system tracks pedestrians in indoor environments by automatically detecting landmarks, pedestrian motion

and human activity by combining data from several sensors. It uses accelerometer readings of the mobile phone to record the number of steps/stairs a person has walked/climbed [15] [21] and accordingly obtain the distance traveled by the person. By utilizing the compass, the direction of the heading changes can be tracked [8]. Using magnetometer readings anomaly context behavior can be detected [1] and finally, the barometer can be used to perceive vertical movement patterns. Additionally, the proposed technique is based on resetting the accumulation of errors by splitting the complete trajectory into independent motion traces.

In order to model indoor human activities, the possible human actions are reduced to five states which correspond with the limitations that indoor obstacles and floor plans impose on pedestrians:

- Turns (T): when the pedestrian changes the heading in his/her route.
- Stationary (Sy): when the person remains in the same location for some time.
- Elevator (E): when the user makes use of an elevator to travel to a different floor.
- Walking (W): when the user moves across the same floor.
- Stairs (St): when the user takes the stairs to go up/down to change floors.

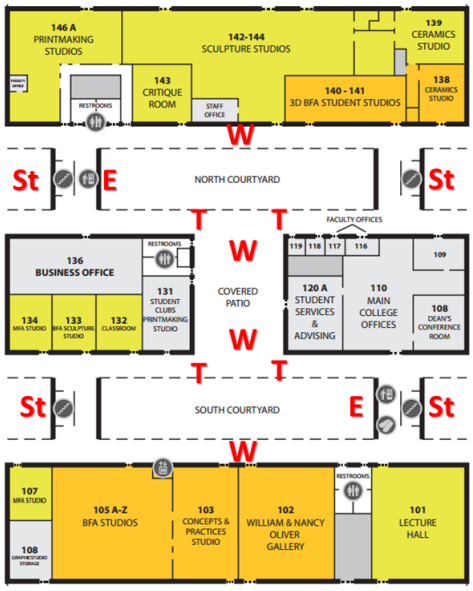

Fig. 1: Real floor plan example.

Consider for example the floor plan of the first floor of the Fine Arts Hall Building at the University of South Florida shown in Figure 1. This real location is basically formed by corridors where users can walk straight (W), corners in which pedestrians make heading changes (T), stairs, where users climb up/down steps (St), and elevators, used to automatically change floors wither up or down (E). Stationary times (Sy) are also considered in which users remain in the same position for a defined period of time.

This set of predictable activities can be translated into identifiable context signatures or landmarks that can be specified using the data gathered by the sensors integrated in the mobile phone. For instance, elevators exhibit a remarkable variation in the magnetic field magnitude added to the pressure variation with the vertical movement. Human walking steps can be identified by a repetitive pattern in the accelerometer raw data, and heading changes can be detected from gyroscope measures. We take advantage of this approach to simultaneously harness sensor-based dead-reckoning and environment sensing.

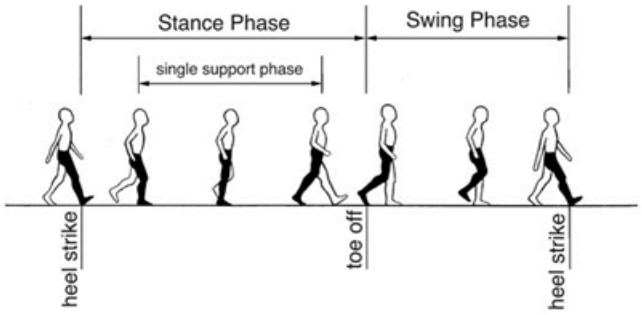

Fig. 2: Human gait cycle.

The proposed system defines a set of rules to automatically detect the activity among the possible states defined above. These rules are based on three key concepts:

1) *Indoor points of interest (POI)*. Multiple tests have shown that certain indoor locations present identifiable signatures on one or more sensing dimensions. These signatures can be detected processing the raw data gathered by the sensors and translated into the real indoor points. This principle is used in our design to detect elevators and corners.
2) *Human body behavior patterns*. Pedestrian actions, like walking, generate repetitive and identifiable patterns that are detectable by the inertial sensors. For instance, human gait is defined as the way that humans walk. Human walk is a bipedal and biphasic forward propulsion in which there are alternate sinuous movements describing a motion cycle. Different segments of the body are involved in the walk activity, mainly in the lower body. The gait cycle begins with the initial contact of the supporting heel on the ground and ends when same heel contacts the ground for a second time. Thus, it can be classified in two phases: stance and swing (Figure 2). Each cycle begins at initial contact with a stance phase (defined as the interval of time in which the foot is on the ground, approximately 60 percent of the gait cycle) and proceeds through a swing phase (defined as the interval of time in which the foot is not in contact with the ground approximately 40 percent of the gait cycle) until the cycle ends with the repetition of the initial contact. The idea of identifying human repetitive patterns is used in this system to detect human steps and climbing stairs.
3) *Elimination of cumulative errors*. The approach of navigation based on landmarks and activities allow us to split the data by activity frames to reduce the

cumulative error of the sensors. Furthermore, every single motion frame is processed to include specific information, such as number of steps/stairs, time in the detected action, distance walked/climbed, elevator direction or turn direction. Thanks to this additional information the system is able to rebuild the user's motion by a sequence of traces. This functionality is useful for pedestrian tracking, indoor positioning, or eventually for participatory floor plan construction.

The architecture of the system consists of four main modules: the data collection module, the motion segmentation module, the activity recognition module, and the activity specification module, as shown in Figure 3. Altogether, these four modules in a sequential manner:

1) Gather raw data from the sensors embedded in the mobile device.
2) Split the complete user's motion into segments using heading and altitude changes as separators or splitters.
3) Classify each segment in an activity using a decision tree based on classification rules.
4) Add additional specifications to each activity creating motion traces that recreate pedestrian indoor movements. This last module applies the specific algorithms (step detection and counting, stairs detection and counting and elevator frame classification).

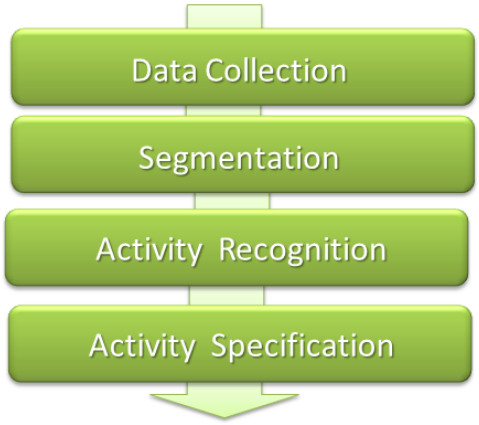

Fig. 3: System architecture.

A. Data Collection Module

The first module in the architecture of the system is responsible for gathering raw data from the various sensors embedded in the user's mobile device. Raw sensor data comes from the inertial sensors, namely the accelerometer, gyroscope, magnetometer, and barometer. These sensors have the advantage of being ubiquitously embedded in most smartphones, have a low-energy footprint, and be always active during the phone operation with the goal of detecting changes in the orientation of the phone or helping in the location of the cell phone. The data collection module was implemented as a service in an Android application. The sensors are sample every $15ms$ ($66.6Hz$). This duty cycle is good enough to detect user's activity and motion details as discussed in the evaluation section.

B. Segmentation Module

The main goal of this module is to split the raw data into independent segments. The data are split into segments using two main events: heading changes, detected by the turn detection algorithm when pedestrians perform a turn in a corner, and altitude changes, identified by the altitude change detection algorithm when the user takes the stairs or an elevator to change floors.

1) Turn Detection Algorithm: Corners are a common occurrence in indoor scenarios and they can be used as splitters to segment the traces. Hence, an important event to be detected in indoor traces is the change on heading directions. Turns can be recognized based on the measurements from the gyroscope and applying the algorithm explained below (Algorithm 1), which is based on significant changes in gyroscope readings. Turns are detected when compass measurements identify heading changes more significant than random variations. The turn detection algorithm performs a calibration routine to compensate the bias introduced by the sensor and applies a filter to reduce the background noise. It sets a threshold over the magnitude of the filtered signal to detect high variations and eventual turns. Finally, the turn's direction is given by the sign of the compass reading with the biggest magnitude.

Algorithm 1 Turn Detection Algorithm

for each $i \in MotionTrace$ **do**
  $G(\alpha) \leftarrow f(g_{\alpha_i})$ *Equation* 1
  $E[g_i] \leftarrow f(g_{x_i}, g_{y_i}, g_{z_i})$ *Equation* 3
end for
$\overline{G(\alpha)} \leftarrow f(G(\alpha), N)$ *Equation* 1
$\overline{G_{calib}} \leftarrow f(\overline{G(x)}, \overline{G(y)}, \overline{G(z)})$ *Equation* 2
for each $i \in MotionTrace$ **do**
  for $j = i - \omega$ **to** $j = i + \omega$ **do**
    $g_i \leftarrow f(E[g_j], \overline{G_{calib}})$ *Equation* 4
  end for
  $\overline{g_i} \leftarrow f(g_i, \omega)$ *Equation* 4
  $G_i \leftarrow f(\overline{g_i}, G_T)$ *Equation* 5
  $T_i \leftarrow f(G_i)$
  Add T_i *to* $TurnInstants[\,]$
end for
for each $T_i \in TurnInstants$ **do**
  $TurnDirection \leftarrow f(T_i, \overline{g_{\alpha_i}}, \omega)$ *Equation* 6
end for
return $[TurnInstants, TurnDirections]$

The algorithm is based on the following steps:

1) Calibration routine: during a trace of movement, gyroscope samples are collected in the three axes $[g_x, g_y, g_z]$. Then, using Equation 1, the mean of the compass values for each axis is calculated.

$$[\overline{G(x)}, \overline{G(y)}, \overline{G(z)}] \quad with \quad \overline{G(\alpha)} = \frac{1}{N} \sum_{i=0}^{N-1} g_{\alpha i} \quad (1)$$

where N means the number of samples used for the calibration routine for one movement trace. Then, Equation 2 is used to calculate the magnitude of the averages, which is considered as bias and used to compensate and shift the compass variations to zero.

$$\overline{G_{calib}} = \sqrt{\overline{G(x)}^2 + \overline{G(y)}^2 + \overline{G(z)}^2} \quad (2)$$

2) Moving average filter: using Equation 3, the energy of the compass samples g_i for every sample i, is computed. Then, Equation 4 applies a moving average filter by estimating the average of the energy in a window of size ω (10 samples) and compensating the bias previously calculated by Equation 2.

$$E[g_i] = \sqrt{g_{x_i}^2 + g_{y_i}^2 + g_{z_i}^2} \quad (3)$$

$$\overline{g_i} = \frac{1}{2\omega + 1} \sum_{j=i-\omega}^{j=i+\omega} (E[g_j] - \overline{G_{calib}}) \quad (4)$$

3) Threshold: defined by Equation 5, it generates a square wave to detect the heading changes based on the high and low levels of the signal.

$$G_i = \begin{cases} G_T & if \ \ \overline{g_i} > G_T \\ 0 & otherwise \end{cases} \quad (5)$$

After multiple empirical tests, the threshold value G_T has been fixed to $1.2rad$. Thus, a high level in the signal G_i means that the compass headings changed more than G_T during an interval of 10 samples around sample i (or $150ms$ with a sampling time of $15ms$).

4) Turn detection: a turn is detected when the square signal G_i shows a period of high level followed by a low level. In other words, a transition from low to high level is detected in G_i $(G_{i-1} < G_i)$ and samples later, a transition from high to low $(G_{i-1} > G_i)$. The turn instant T_i is estimated in the center of this G high level period.

5) Turn direction: once a turn is detected for sample i, in order to determine its direction it is necessary to study the sign of the compass reading with the biggest magnitude. The function $MainComponent$ (Equation 6) returns the biggest gyro component (maximum absolute value) in a window of size w with center in the detected turning instant T_i.

$$MainComponent(i) = max\left\{\overline{g_{xi}}, \overline{g_{yi}}, \overline{g_{zi}}\right\}$$
$$with \ \ \overline{g_{\alpha i}} = \frac{1}{2\omega + 1} \sum_{i=j-\omega}^{i=j+\omega} abs(g_{\alpha i}) \quad (6)$$

According to the sensor's coordinate system in the Sensors Android API and the standard mathematical definition of positive rotation, a rotation is positive when follows the counterclockwise direction. It means that, an observer looking at a device positioned on the origin from some positive location on the x, y or z axes would report positive rotation if the device is rotating counterclockwise. Therefore if the $MainComponent$ value for the detected turn sample is positive, a turn to the left has been performed. Otherwise, if the sign is negative, a turn to the right has been performed.

As an example, the top part of Figure 4 shows the values of the signals as calculated by Equations 3, 4 and 5 in the turn detection algorithm while the bottom part of the figure shows the components of the compass readings as computed by Equation 6. The trace motion in the example includes a turn to the left between samples $95-110$, a right turn between samples $200-215$ and a left turn between samples $275-290$.

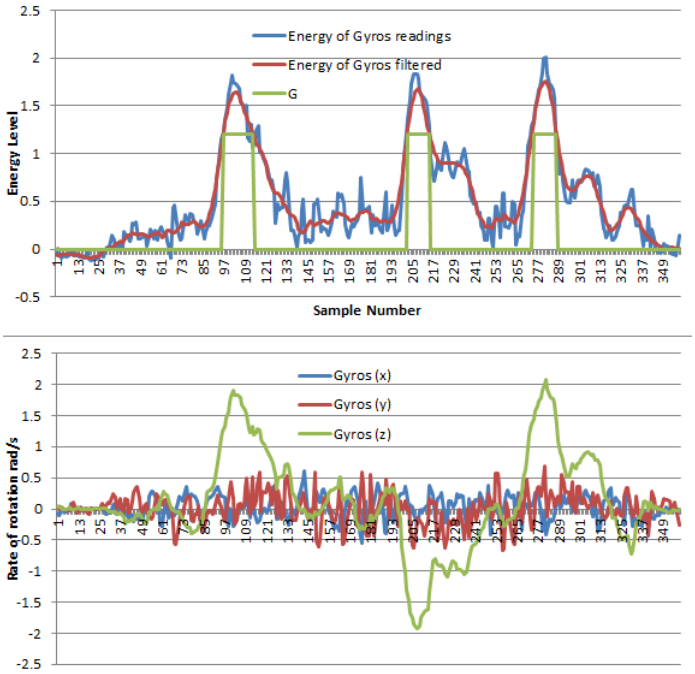

Fig. 4: Turn detection algorithm signals.

2) Altitude Change Detection Algorithm: Altitude changes are critical to detect activities that involve a change of floor. For instance, it is useful to differentiate between walking in flat floors or stairs. Similarly to heading changes, altitude changes can be used as splitters to segment the raw data. The altitude change detection algorithm (Algorithm 2) is based on significant changes in the air pressure raw data acquired by the barometer.

Algorithm 2 Altitude Change Detection Algorithm

```
for each i ∈ MotionTrace in windows i + N do
    for i to i + N do
        P_i ← f(p_{i-1}, p_i, β) Equation 9
        P(t[i]) ← f(P_i) Equation 8
    end for
    f(P(t[i]) ← f(P(t[i]), N) Equation 8
    AC_i ← f(P(t[i + 1]), P(t[i]), StDev) Equation 7
    Add AC_i to AltitudeChanges[ ]
end for
return AltitudeChanges[ ]
```

Altitude changes are recognized when the barometric pressure values vary more significantly than due to random oscillations, as given by Equation 7.

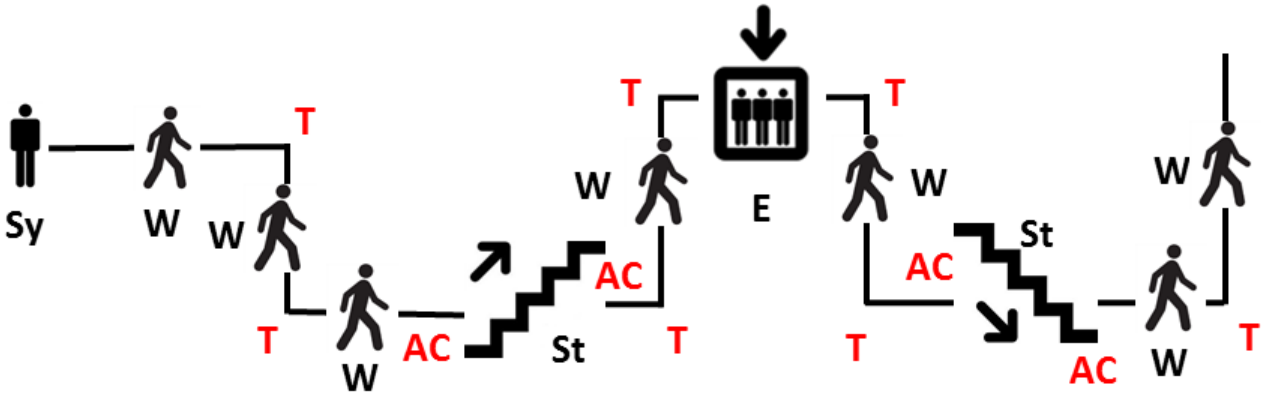

Fig. 5: Example of a pedestrian trace split by turns and altitude changes.

$$\overline{P(t[i+1])} - \overline{P(t[i])} \geq StDev(P) \tag{7}$$

in which the first term represents the variation of the average between two consecutive intervals. In Equation 7, $\overline{P(t[i])}$ denotes the average of the low pass filter values for the air pressure readings over a $t[i]$ time period, as calculated by Equation 8, N is the number of samples included in the period t, and P_i is the low pass filter signal of the pressure readings p_i.

$$\overline{P(t[i])} = \frac{1}{N} \sum_{i=0}^{N-1} P_i \tag{8}$$

The filter has been implemented using a discrete implementation of a basic RC low-pass filter, as show in Equation 9, with a smoothing factor of $\beta = 0.9$ in order to obtain a smoother form of the signal, remove the short-term oscillations and maintain the longer-term trend.

$$P_i = LPF[p_i] = \beta\ p_{i-1} + (1-\beta)\ p_i \tag{9}$$

The second term in the detection condition of the algorithm (Equation 7) identifies the pressure random oscillations. It is detected based on the standard deviation estimated for the barometric pressure sensor in resting conditions (not moving), $0.05 mbar/sec$. This value was obtained from the quality sensors test shown in the evaluation section of this paper.

In those time intervals in which a significant variation is detected (AC_i), the trend of the pressure signal is analyzed to generate a square wave to represent the altitude changes ($+1$ and -1, respectively). It is worth noting that, a notable increase in the pressure signal means a decrease in altitude and vice versa.

Figure 6 shows an example of the signals in the altitude change detection algorithm. The figure on the top left shows the pressure raw data; the one on the top right shows the pressure measurements after being computed by the low pass filter as calculated by Equation 9. The figure on the bottom left shows the difference of averages between two consecutive intervals as computed by Equation 7. Finally, the figure on the bottom right shows the altitude changes detected. In the example, a positive altitude change was detected in the interval between samples number 60 and 250, and a period of negative altitude, or going down, was detected in the interval between samples 540 and 740.

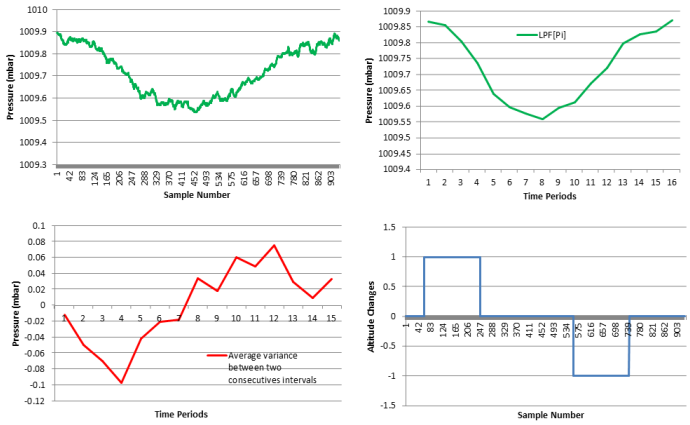

Fig. 6: Altitude change detection algorithm signals.

Once detected, turns and altitude changes are used in the proposed solution to split the continuous motion traces in a sequence of independent segments. This trace segmentation technique reduces the error accumulation introduced by the inertial sensors. This approach contrasts with classical relative navigation techniques, such as dead reckoning in which the new location of a user is estimated using the previous location, the distance traveled and the direction of motion.

Figure 5 shows an example of an indoor motion trace and sequence of activities split by Turns (T) and Altitude Changes (AC).

C. Activity Recognition Module

The aim of this module is to define a set of rules that automatically allows the system to detect the segment activity among the possible indoor states defined above (stationary,

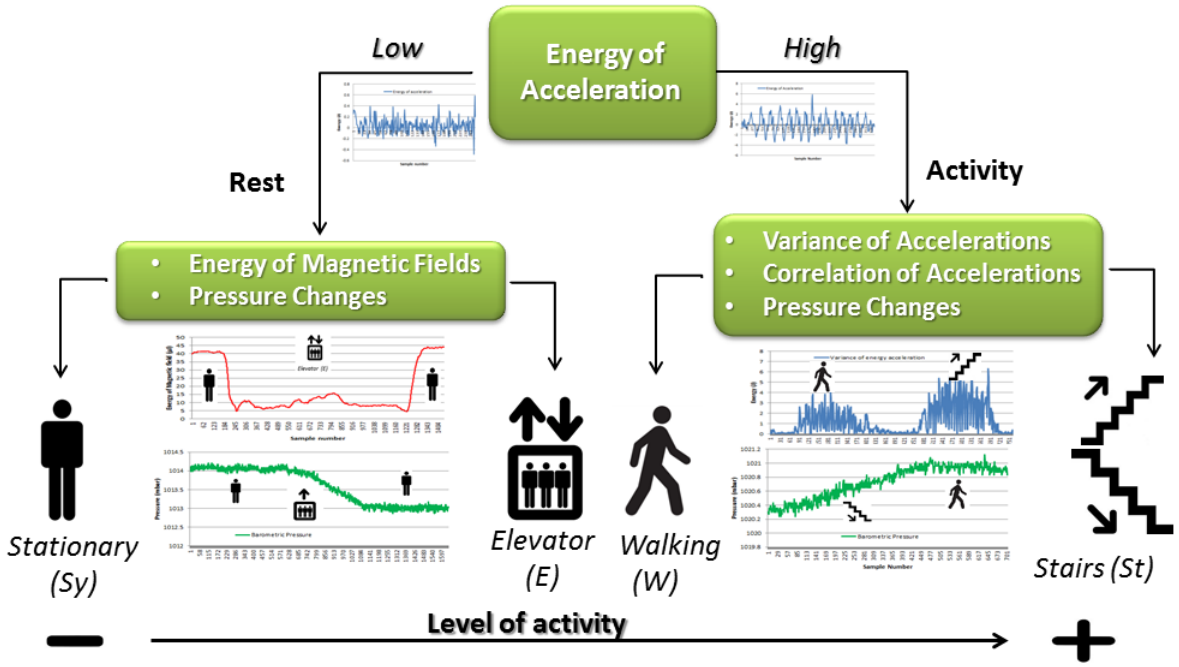

Fig. 7: Decision tree to classify the possible indoor states.

elevator, walking or stairs). This module receives the segments of motion generated by the previous module and processes the sensor data to classify the segments and estimate the activity performed. Figure 7 shows the decision tree that has been defined to classify the possible states.

1) Activity or Rest: At the very top of the tree, the first decision to make is to decide whether the user was active or at rest. The first decision is to differentiate between movements based on the magnitude level of acceleration, a process similar to the one used in the turn detection algorithm. After filtering the signal to make it smoother, the first decision is to differentiate between movements based on the magnitude level of acceleration. Fixing a threshold over the acceleration energy leads to estimate periods of activity and, complementary, segments of rest. Analogous, other inertial sensor signals can be used to detect the activity subclasses in the decision tree, such as the magnitude of the magnetic fields, the pressure changes, the variance of accelerations, or the correlation of the accelerations.

2) Elevator (E): According to the decision tree, if the user is at rest is because he or she is either stationary or in an elevator. Similar to cars or planes, elevators behave like a Faraday shield presenting a unique magnetic field pattern that makes them distinguishable with accuracy. Since a typical elevator is a structure formed by conducting material, it blocks non-static electric fields and external static. The different values for the magnetic field energy coming from the outside and the inside of the elevator show a notable difference, thus this transition of states is easily identifiable, as shown in Figure 8.

There are two additional details that have to be distinguished in the elevator motion segments: the direction of motion and the estimation of the number of floors traveled.

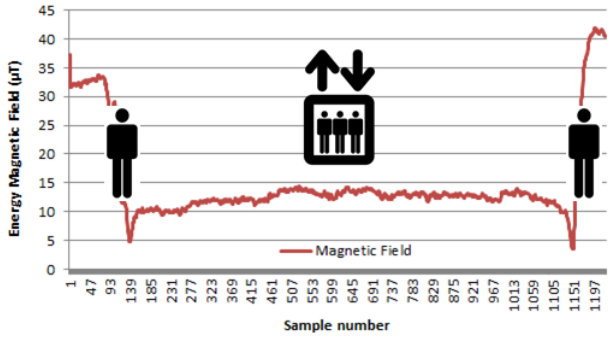

Fig. 8: Transition of magnetic field energy outside and inside an elevator.

The identification of the elevator direction can be estimated based on the energy of the acceleration measurements when the elevator starts and stops its travel. These events produce a pattern of acceleration peaks in the elevator segment and studying its order of appearance the elevator motion can be classified. If the pattern is a positive peak followed by a negative peak, a travel in the up direction was performed (see Figure 9). If the sequence shows a negative peak followed by a positive one, the elevator traveled down.

The number of floors traveled can be estimated in two different ways. First, it can be estimated with the aid of the displacement duration of the travel inside the elevator, considering the number of samples between the acceleration peaks detected. Second, it can also be estimated using the barometric sensor included in recent smartphones. With the barometer raw data, the detection of pressure changes helps enormously in the

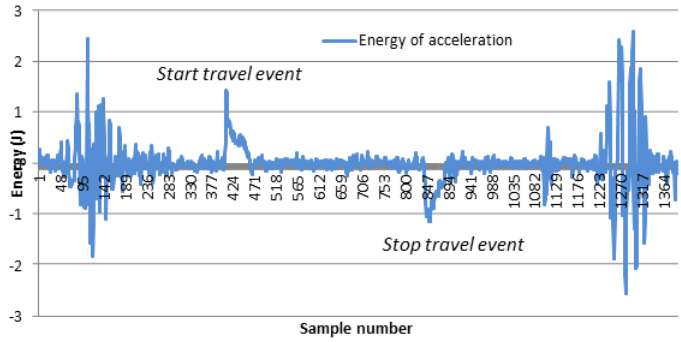

Fig. 9: Energy of acceleration values when the elevator travels in the up direction.

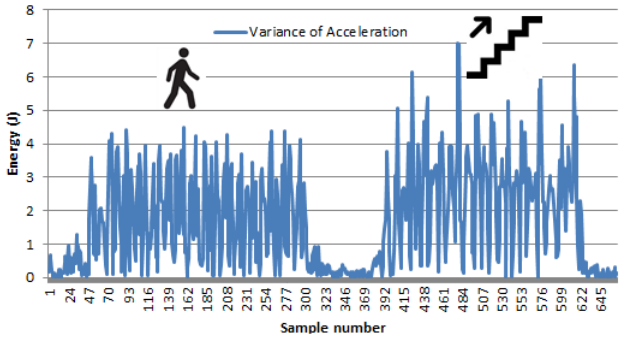

Fig. 11: Variance of energy acceleration in walking, stationary and stairs states.

detection of altitude changes, thus in the recognition of elevator direction travels. As shown in Figure 10, an elevator traveling up means a notable decrease in the air pressure measurements. The number of floors traveled can be estimated considering the period of time during which the barometer shows an abnormal variation.

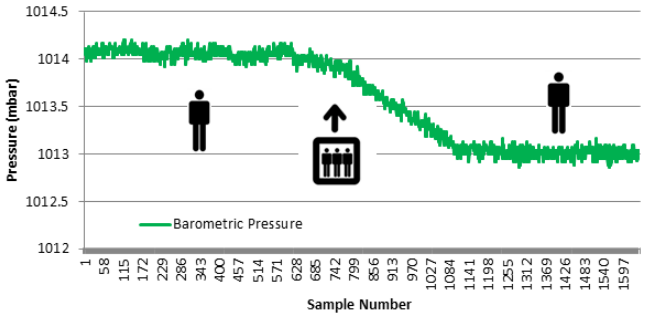

Fig. 10: Air pressure variations when the elevator is traveling in the up direction.

3) Stationary (Sy): Stationarity or being at rest includes sitting or standing with no displacement performed during a period of time. Since elevator and stationary states are the only two possible states in the left branch of the decision tree, if a segment is not classified as elevator, it will be considered as a stationary segment.

4) Walking (W) and Stairs (St): The second possibility at the very top of the tree is when the user is active. Once the rest states have been discarded using the energy of the acceleration, it is necessary to differentiate the active segments between stairs and walking cases. The initial observation is that when pedestrians are taking the stairs, the variance of the acceleration is broader than in the walking case. This can be seen in Figure 11, which shows the variance of the acceleration's energy in a Walking (W) - Stationary (Sy) - Stairs (St) motion sequence. The correlation between the acceleration signals in the motion's direction (y axis) and the direction of gravity (z axis) is a good indicator to separate stairs from walking. Furthermore, the measurements show that taking the stairs down (helped by the acceleration of gravity) involves higher motion intensity than going up.

In addition, for mobile devices that include a barometric sensor, the detection of altitude changes can be useful to find the different active states and the direction of movement during stairs periods. As shown in Figure 12, the trend of air pressure measurements helps in the identification of the type of activity performed. It can be an accurate signature for the detection of walking the stairs or walking flat.

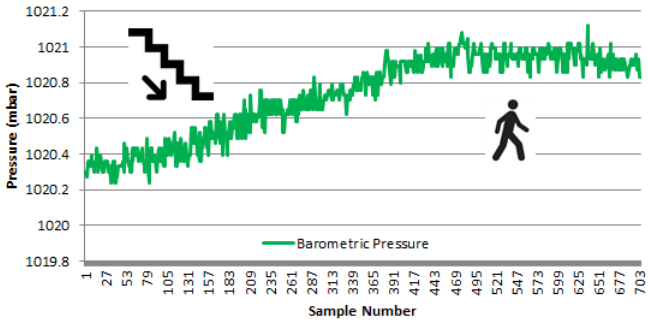

Fig. 12: Values of air pressure in stairs and walking states.

Combining these three processed signals (variance and correlation of accelerations and pressure changes) and using them as indicators, the active states can be detected, including the direction of the user in the stairs segments. On the one hand, a segment with high variance in the acceleration, a high yz correlation in the acceleration and notable increases in air pressure will be detected as going down the stairs. On the other hand, a high variance, a medium value of yz correlation and decreasing changes in air pressure will be detected as going up the stairs. Finally, a low level in the variance and the yz correlation, combined with a stable level in the air pressure readings will classify the segment as walking.

D. Activity Specification Module

Once the activity recognition module has identified the activity in a segment, some activities, like walking or stairs, require of additional algorithms to enable a complete tracking of the user. In this section we describe additional algorithms to detect steps and stairs and therefore being able to estimate the total distance traveled by the user.

1) Step Detection and Counting Algorithm: Thanks to the detection of cycles in the data gathered by the accelerometer (swing and stance phases) caused by the repeated patterns or events in motion of walking, it is possible to count the number of steps, and therefore obtain and estimation of the distance traveled. Based on the experiments, the effect of walking on the magnitude of the acceleration vector is independent from the phone orientation and tilt. Consequently, our step counting algorithm is designed using the magnitude of the acceleration, making this approach for distance estimation independent from the placement of the mobile phone (messaging in hands, calling in user's ear or swinging in the pocket). The step detection and counting algorithm (Algorithm 3) performs a calibration routine that compensates the bias introduced by the sensor and applies an average filter to reduce the background noise. A double threshold over the magnitude of the filtered signal is used to detect the stance and swing phases in the human gait. Finally, a step is detected when a transition between stance and swing phases is recognized.

Algorithm 3 Step Detection and Counting Algorithm

for each $i \in MotionTrace$ **do**
  $A(\alpha) \leftarrow f(a_{\alpha i})$ *Equation* 10
  $E[a_i] \leftarrow f(a_{x_i}, a_{y_i}, a_{z_i})$ *Equation* 12
end for
$\overline{A(\alpha)} \leftarrow f(A(\alpha), N)$ *Equation* 10
$\overline{A_{calib}} \leftarrow f(\overline{A(x)}, \overline{A(y)}, \overline{A(z)})$ *Equation* 11
for each $i \in MotionTrace$ **do**
  for $j = i - \omega$ **to** $j = i + \omega$ **do**
    $a_i \leftarrow f(E[a_j], \overline{A_{calib}})$ *Equation* 13
  end for
  $\overline{a_i} \leftarrow f(a_i, \omega)$ *Equation* 13
  $A_{1_i} \leftarrow f(\overline{a_i}, A_{T_1})$ *Equation* 14
  $A_{2_i} \leftarrow f(\overline{a_i}, A_{T_2})$ *Equation* 15
  $S_i \leftarrow f(A_{1_i}, A_{2_i}, \omega)$
  Add S_i *to* $StepDetections[\,]$
end for
return $StepDetections[\,]$

The algorithm implemented for step detection and counting consists of the following steps:

1) Calibration routine: during a segment of movement, samples of linear acceleration $[a_x, a_y, a_z]$ are collected to estimate the mean of the acceleration using Equation 10.

$$[\overline{A(x)}, \overline{A(y)}, \overline{A(z)}] \quad with \ \overline{A(\alpha)} = \frac{1}{N}\sum_{i=0}^{N-1} a_{\alpha i} \tag{10}$$

where N is the number of samples used in the calibration routine for one segment. Then, the energy of the averages is calculated using Equation 11, which is considered as bias to compensate.

$$\overline{A_{calib}} = \sqrt{\overline{A(x)}^2 + \overline{A(y)}^2 + \overline{A(z)}^2} \tag{11}$$

2) Mean of accelerations: compute the energy of the acceleration $E[a_i]$ for every sample i, as shown in Equation 12.

$$E[a_i] = \sqrt{a_{x_i}^2 + a_{y_i}^2 + a_{z_i}^2} \tag{12}$$

Use Equation 13 to estimate the average of the energy in a window of size ω (10 samples) and compensate for the bias as calculated by Equation 11.

$$\overline{a_i} = \frac{1}{2\omega + 1}\sum_{j=i-\omega}^{j=i+\omega} (E[a_j] - \overline{A_{calib}}) \tag{13}$$

3) Thresholds: a first threshold A_{T_1} is applied according to Equation 14 to detect high accelerations during the swing phase.

$$A_{1_i} = \begin{cases} A_{T_1} & if \ \ \overline{a_i} > A_{T_1} \\ 0 & otherwise \end{cases} \tag{14}$$

A second threshold A_{T_2}, as defined by Equation 15, is used to detect the walking stance phase.

$$A_{2_i} = \begin{cases} A_{T_2} & if \ \ \overline{a_i} < A_{T_2} \\ 0 & otherwise \end{cases} \tag{15}$$

A_{T_1} and A_{T_2} are symmetric values respect to 0 and fixed heuristically to $0.5m/s^2$, meaning that the acceleration varies more than A_{T_1} or less than A_{T_2} during an interval of $150ms$ or 10 samples.

4) Step detection: a step S_i is identified in the sample i when a swing phase ends and a stance phase starts. For a step detection two sequential conditions must be accomplished:
   a) a change from high to low acceleration ($A_{1_{i-1}} > A_{1_i}$) is detected in the square wave generated after applying the threshold A_{T_1} (Equation 14), and simultaneously
   b) there is at least one detection of a low level of acceleration in a window of size ω ahead of the current sample i, i.e., $min(A_{2_{i:i+\omega}}) = A_{T_2}$ in the square wave generated after applying the threshold A_{T_2} (Equation 15).

5) Finally, the array with the step samples is iterated to obtain the average time between steps, which could be useful to estimate the step rate or the velocity of displacement of the person.

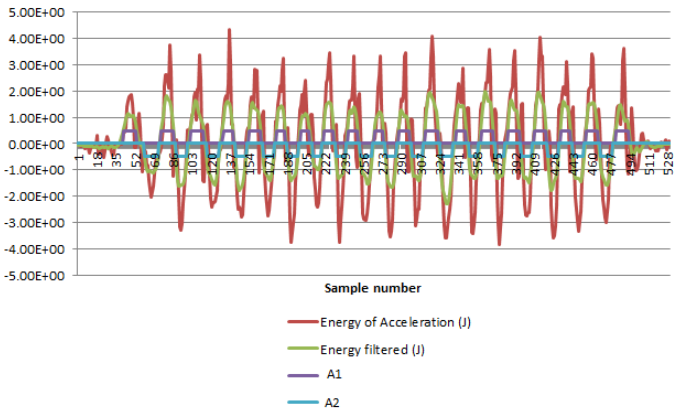

Fig. 13: Step detection and counting algorithm signals.

Figure 13 shows the signals as a result of applying Equations 12, 13, 14 and 15 in the step detection and counting algorithm for a basic walking example. Once the number of steps is obtained, the total distance walked can be directly estimated considering the stride length of each step to be constant and with a value of $0.74m$ [2].

2) Stairs Detection and Counting Algorithm: The algorithm in this section estimates the number of steps while using the stairs and the total distance traveled by the user. This algorithm (Algorithm 4) also performs a calibration routine to compensate for the bias and applies a low pass filter over the energy of the accelerations to reduce the background noise. Finally, it looks for peaks in the signal and applies a guard factor to discard minor variations and detect only the peaks that are eligible as stairs.

Algorithm 4 Stair Detection and Counting Algorithm

for each $i \in MotionTrace$ **do**
  $A(\alpha) \leftarrow f(a_{\alpha i})$ *Equation* 10
  $E[a_i] \leftarrow f(a_{xi}, a_{y_i}, a_{zi})$ *Equation* 12
  $A_i \leftarrow f(E[a_{i-1}], E[a_i], \alpha)$ *Equation* 16
end for
$\overline{A(\alpha)} \leftarrow f(A(\alpha), N)$ *Equation* 10
$\overline{A_{calib}} \leftarrow f(\overline{A(x)}, \overline{A(y)}, \overline{A(z)})$ *Equation* 11
for each $i \in MotionTrace$ **do**
  $\overline{A_i} \leftarrow f(A_i, \overline{A_{calib}})$ *Equation* 17
  $P_i \leftarrow f(\overline{A_i}, \overline{A_{i-1}}, \overline{A_{i+1}})$ *Equation* 18
  Add P_i *to* $Peaks[\,]$
end for
for each $i \in Peaks[\,]$ **do**
  $C \leftarrow f(\overline{A_i}, M, G)$ *Equation* 19 *and* 20
  $St_i \leftarrow f(C, \overline{A_i})$
  Add St_i *to* $StairDetections[\,]$
end for
return $StairDetections[\,]$

The algorithm implemented for stairs detection and counting based on peak detection consists of the following steps:

1) Calibration routine: during a segment, samples of linear acceleration $[a_x, a_y, a_z]$ are collected to estimate the mean in the acceleration. The energy of the average is estimated and considered as bias to compensate. The same Equations 10 and 11 are used here.
2) Energy of acceleration: the simplest way to produce useful data out of the three components of the sensor is to take the magnitude of the acceleration vector. It computes the energy of the acceleration $E[a_i]$ for every sample a_i using Equation 12.
3) Low pass filter and bias compensation: low-pass filters provide a smoother form of the signal removing the short-term fluctuations. A_i is the discrete low pass filter signal of the acceleration energy readings $LPF(E[a_i])$. It has been applied using a discrete-time implementation of a simple RC low-pass filter as show below in Equation 16, with a smoothing factor of $\alpha = 0.9$.

$$LPF(E[a_i]) = \alpha\ E[a_{i-1}] + (1-\alpha)\ E[a_i] \quad (16)$$

To compensate the bias, the value $\overline{A_{calib}}$ is removed for all the energy filtered samples using Equation 17.

$$\overline{A_i} = A_i - \overline{A_{calib}} \quad (17)$$

4) Peak Detection: a peak P_i is detected at a sample i if during the Ω previous samples ($\Omega = 5$, meaning $75ms$) the $backwardSlope$ of the current sample is positive and in the next sample the $forwardSlope$ becomes negative. These two functions are detailed in Equation 18.

$$backwardSlope(\overline{A_i}) = \overline{A_i} - \overline{A_{i-1}}$$
$$forwardSlope(\overline{A_i}) = \overline{A_{i+1}} - \overline{A_i} \quad (18)$$

5) Stairs detection: the energy signal is traversed by a buffer of a fixed number of samples. In this implementation, the buffer length is 100 samples equal to $1.5s$. For every set of samples in the buffer, $peakMean$ (Equation 19) is calculated estimating the energy of the detected peaks P_i.

$$peakMean = \frac{1}{M} \sum_{i=0}^{N-1} \overline{A_{P_i}} \quad (19)$$

where M is the number of samples detected as peaks and possible stairs. This value is multiplied by a guard factor ($G = 0.6$) to avoid the detection of false peaks, and the new value C (Equation 20) is the threshold for each set of samples in the buffer.

$$C = G\ * peakMean \quad (20)$$

The peaks detected in each buffer have effect on the responsiveness of the algorithm changing the value of the threshold C. The final step of the algorithm is to detect stairs St_i by iterating over $\overline{A_i}$ and detecting the peaks that are above the threshold.

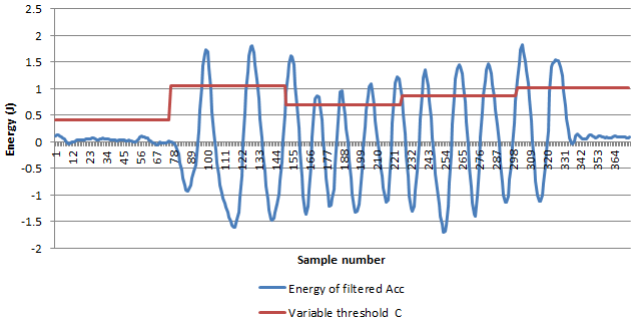

Fig. 14: Stairs detection and counting algorithm signals.

Figure 14 shows the signals taking part in the stairs detection and counting algorithm applied in a motion test in which a user was going down the stairs. The total altitude climbed can be estimated considering the number of steps and the maximum stair riser heights. This value is regulated and fixed to $7inches$ ($178mm$) by the International Building Code (IBC). The IBC is a model building code which has been adopted throughout most of the United States. It is developed and maintained by a standards organization independent of the jurisdiction responsible for enacting the building.

IV. System Implementation

The system was implemented in two main parts as shown in Figure 15: the mobile application (called SensorApp), which is in charge of gathering the data from the mobile device and displaying the values on the screen; and the server, which applies the activity recognition decisions and the data processing algorithms, saves the traces, provides tools to introduce the experiments for testing, and displays the evaluation results.

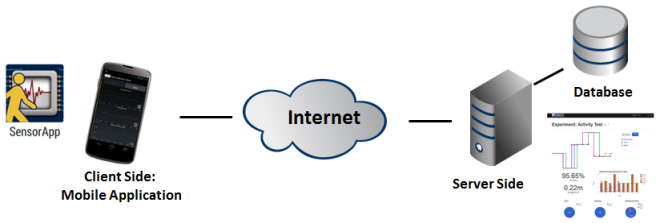

Fig. 15: System implementation architecture.

A. Client Side: Mobile Application

An Android application was developed to acquire the raw data from the sensors embedded in the smartphone. SensorApp makes use of the Android Sensor API and provides two screens to visualize the data, as shown in Figure 16. Both screens show the raw data collected from the embedded inertial sensors available in the device: accelerometer, gyroscope, magnetometer and barometer. The first screen shows several text views where the values of the raw data collected are displayed. The second screen represents the sensor values in graph format where the x axis shows time and y axis can be: m/s^2 for the acceleration, rad/s for the rotation, μT for the magnetic field, or $mbar$ in the case of the air pressure. The y axis scale is adapted to the current measurements to give a better resolution of the values. Additionally, the mobile application is in charge of transmitting the information packets to the server using a web service when the motion test finishes.

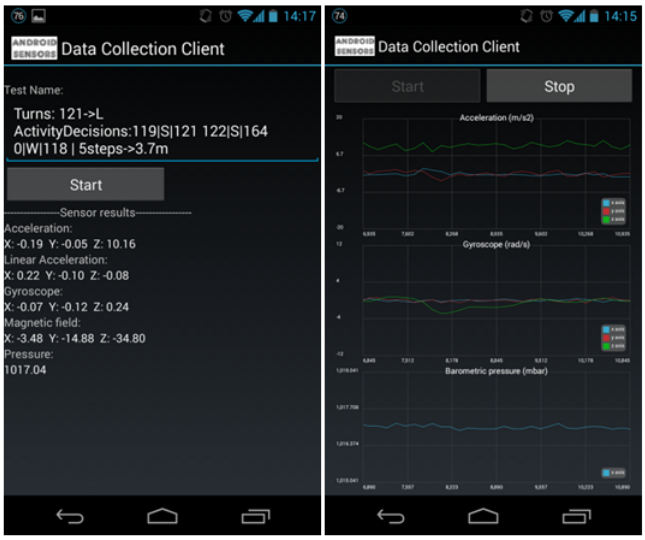

Fig. 16: Mobile application screens.

B. Server Side

The server side has been designed as the main module of the system. It aims to be a test environment to support and simplify the system's implementation and the evaluation process. It allows to create experiments for different real scenarios and to compare them with indoor motion sequences obtained from the tests performed. Thus, it automates much of the testing process and makes simpler the evaluation of the results. The server performs the following tasks:

- Receiving data. It includes a web service to receive the raw data sent by the mobile device at the end of every single test.
- Data processing algorithms. The different modules explained in the system design are accomplished by several synchronized threads. These algorithms will return the activities detected and their details.
- Storage. The system saves the information returned by the processing algorithms for future references, queries and evaluations. It also stores the experiments to be tested.
- Experimentation. The server provides a graphical interface to create experiments, i.e., a sequence of turns, steps, etc. The experiments are then supposed to be performed as such using the mobile device identifying them with the ID created by the server. Upon completion of the data collection, the mobile device sends the data to the server, which uses the experiment ID to compare the experiment generated (ground truth) with the results of analyzing the raw data.
- Visualization. The server includes a web application to visualize and represent the results of the experiments, so they are easy to read and analyze, as shown in Figure 17.

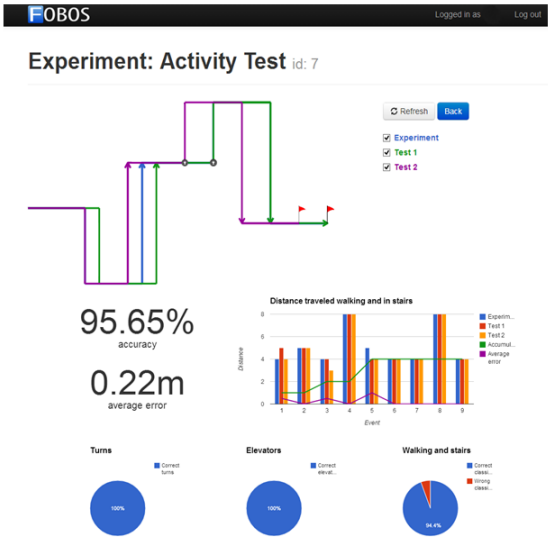

Fig. 17: Server web application screenshot.

V. EVALUATION

In this section, the performance of the proposed system is evaluated with several basic tests and complete motion traces involving different activities. The section starts describing the methodology applied, followed by a set of quality sensor tests for the inertial sensors and presenting the evaluation of all the modules and algorithms involved in the system. It concludes with the discussion of the test results.

A. Methodology

The client side of the system was implemented using the Android platform in two different devices: the Samsung Galaxy SII I777 and the LG Nexus 4 E960, which are equipped with basic inertial sensors such as accelerometer, gyroscope, magnetometer and barometer. The set of experiments were carried out in different buildings in the campus of the University of South Florida, with plenty of corridors, corners, stairs and elevators. The evaluation requires three steps:

1) Create the experiment in the graphical interface of the server specifying the sequence of motion to be performed by the user.
2) Perform the test with the mobile application identifying the experiment to be tested.
3) Compute the results and show them to the user.

A total of 200 motion traces were collected while the user was carrying the phone in his hand, in texting position. Initially, the quality of the sensors is analyzed and evaluated. Also, the convenience of the chosen sampling frequency is demonstrated. Then, the algorithms detecting turns and altitude changes are evaluated. These algorithms allow the segmentation of the original traces. The evaluation continues studying the results of the activity recognition module in charge of classifying the segments in one of the possible states. The results for the specification module are also shown. They include the results of the algorithms for detecting and counting steps and stairs. Moreover, the counting steps algorithm is compared with the step counting hardware-based solution included in the latest version of the Android platform. Finally, the combined tracking accuracy, when applying the complete architecture of the system, is presented.

B. Data Collection Results

1) Inertial Sensors Evaluation: This section performs an evaluation of the inertial sensors considered as data sources in the system. This evaluation is carried out to study the long term errors introduced by the sensors and to consider the need of calibration to obtain more reliable outputs. The set of inertial sensors embedded in the Samsung SII are a 3-axes accelerometer, a gyroscope, and a magnetometer. In addition to all these sensors, the LG Nexus 4 also includes a barometric pressure sensor.

To analyze the accuracy and the behavior of the sensors, tests are performed with the devices stationary on a table. In these error analysis tests, the sensor samples are recorded during a period of $15s$ with the phone lying flat with its back on the table, i.e., with the z axis pointing to the sky. The phone is stationary in order to prevent any force other than gravity from affecting the output.

The ***accelerometer*** measures the acceleration in three axes in m/s^2. It outputs the acceleration applied to the device by measuring forces exerted to the sensor. The measured acceleration is always influenced by the force of the earth's gravity, as shown in Equation 21 and considered as bias to compensate.

$$a_d = -g - \sum \frac{F}{m} \tag{21}$$

where a_d is the acceleration applied to the device, g is the force of gravity, F is the force acting on the device, and m is the mass of the device. The sign $\sum$ represents the sum of the x, y, and z axes.

As a result, when the device is not undergoing any acceleration, the accelerometer output should read $0m/s^2$ in the x and y axes, and a negative Earth's gravity of $9.81m/s^2$ in the z axis. The accelerometer test results, included in Table I, show the high variability of the magnitude of the acceleration measured in the stationary position, in particular, in the case of the Samsung smartphone with $0.13m/s^2$. This is equivalent to more than one percent of the total acceleration, which, with time, could potentially generate a large error. These errors are compensated in the system implementing the initial calibration methods described in Section III.

The ***gyroscope*** readings are in radians per second and measure the rate of rotation around the x, y, and z axes. Rotation is positive in the counterclockwise direction. When the device is at rest on top of a table, the gyroscope values should read a magnitude of $0rad/s$. Table I shows the magnitude of the offset or bias introduced by the rotation sensor, calculated as the standard deviation of the gyroscope outputs when it is not undergoing any rotation. To calculate the angle of rotation error α (rad), the deviation for the angular speed of the gyroscope ω (rad/s), should be integrated over time t as shown in Equation 22.

$$\alpha = \sum(\omega * \Delta t) \tag{22}$$

During a $15s$ test, an error of $0.2365rad$ was estimated for the Samsung Galaxy SII and $0.0102rad$ for the LG Nexus 4. Since the expected error average is zero, the calculated values are approximately equal to the bias error of the sensor. This error is compensated in the gyroscope algorithm in the system performing the initial calibration methods described in Section III.

The ***magnetometer*** measures the strength of the ambient magnetic field in micro-Tesla (μT) in the x, y, and z axes. Ideally, a magnetometer completely isolated should measure a magnitude of $0\mu T$. Table I shows the statistical results from the magnetic field signals collected during a test of $15s$ with both cell phones in a stationary position. Since the measurements are not centered at $0\mu T$, because the magnetometer is not isolated, the average standard deviation is the only useful value to analyze the sensitivity of the sensor. The tests show a rather high standard deviation of $0.403\mu T$ and $0.275\mu T$ for the Galaxy SII and the Nexus 4, respectively. Due to this deviation, calibration and signal filtering routines, similar to the ones described in Section III for the accelerometer and

TABLE I: Inertial sensors test results.

	Accelerometer (m/s^2)		**Gyroscope** (rad)		**Magnetometer** (μT)		**Barometer** $(mbar)$	
	Average	*Std deviation*	*Average*	*Std deviation*	*Average*	*Std deviation*	*Average*	*Std deviation*
Samsung Galaxy SII	9.42610	0.13240	0.01858	0.01577	-51.05188	0.40299	-	-
LG Nexus 4	10.35168	0.04543	0.00099	0.00068	-56.45019	0.27505	1014.69	0.050086

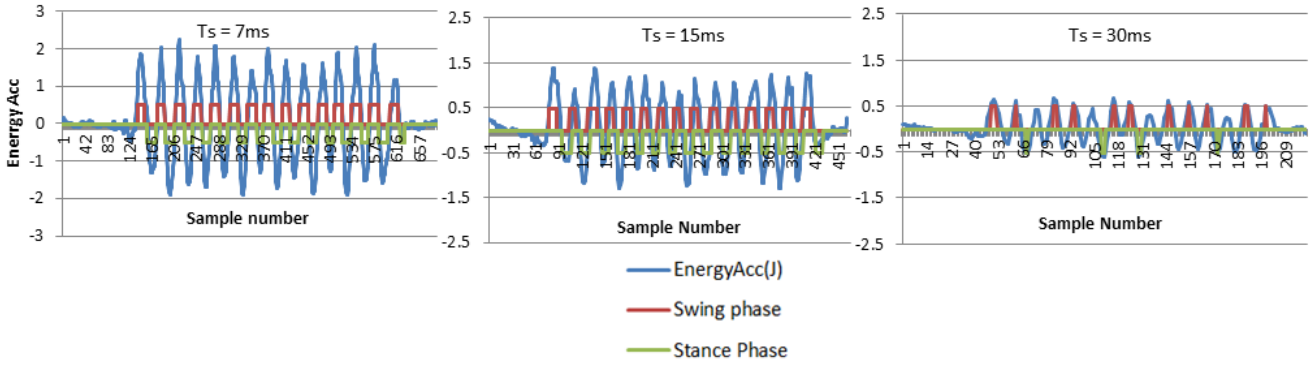

Fig. 18: Output signals for sampling rate test.

gyroscope, are applied to the magnetometer raw data to reduce and compensate the eventual sensor errors in the system.

The barometric air pressure sensor or ***barometer*** is classified as an environmental sensor. It measures the ambient air pressure in $mbar$. Recognizing changes in the barometric air pressure are useful to recognize altitude changes. Similar to the rest of the sensors, a test was performed to estimate the bias error introduced by the barometric sensor embedded in the LG Nexus 4 (the Samsung SII has no barometer). The standard deviation estimated in this test (see Table I) is used in the altitude change detection algorithm of the systems to fix a threshold to eliminate random pressure oscillations.

2) Sampling Rate Test: In all the experiments, a sampling interval of $15ms$ was used to obtain raw data from the sensors. This sampling interval was found empirically after several tests since it provided enough data to determine the activity of the user without consuming more energy than necessary. To get to that conclusion, an experiment performing a walking activity was repeated using sampling intervals of $7ms$, $15ms$, and $30ms$. It is worth mentioning that the walking activity is the most demanding final state in terms of sample resolution among the ones considered in the system. Thus, it imposes the maximum value for the sampling parameter. Figure 18 shows the output signals of the walking segment sampled at those intervals. As it can be seen, the $15ms$ sampling interval offers the best trade off between accuracy and overhead. As the figure shows, the resolution and magnitude of the energy signals decreases as the sampling interval increases, making it more difficult to the step detection and counting algorithms to work correctly. In fact, it can be seen how the number of steps is correctly determined sampling at $7ms$ and $15ms$ (15 steps) while the algorithms only count 11 steps at $30ms$. At the same time, the figure shows that it is not necessary to sample the signal so frequently, as an interval of $15ms$ offers the same final results.

C. Segmentation Results

This section evaluates the algorithms for detecting the events considered as separators in the segmentation module of the system: heading changes, when pedestrians perform a turn in a corner, and altitude changes, when the user takes stairs or an elevator to change floors. These turn detection and altitude change algorithms allow the segmentation of the original complete motion trace into independent segments.

1) Turn Detection Test: Table II summarizes the results of the heading change detection algorithm for six different sequences with two, three, and four heading changes in which L means a turn to the left in a corner and R means a turn to the right. The sequence of turns are detailed in Figure 19. Each sequence was repeated 5 times for a total of 30 experiments and 90 turns. As it can be seen from the table, out of 90 turns in 30 experiments, the algorithm only made one mistake for an accuracy of 98.8%.

TABLE II: Turn detection test results.

Turn Detection Test	**Test 1 result**	**Test 2 result**	**Test 3 result**	**Test 4 result**	**Test 5 result**	**Number of Errors**
Sequence 1 (LL)	LL	LL	LL	LL	LL	0
Sequence 2 (RR)	RR	RR	RR	RR	RR	0
Sequence 3 (LRL)	LRL	LRRL	LRL	LRL	LRL	1
Sequence 4 (RLR)	RLR	RLR	RLR	RLR	RLR	0
Sequence 5 (LRRL)	LRRL	LRRL	LRRL	LRRL	LRRL	0
Sequence 6 (RLLR)	RLLR	RLLR	RLLR	RLLR	RLLR	0

2) Altitude Change Detection Test: The altitude change detection algorithm was evaluated using six different altitude change sequences, involving one, two, and three changes, respectively. The sequences of altitude variation are shown in Figure 20, where U means an interval of time going up, and D a time going down. Each experiment was repeated 5 times,

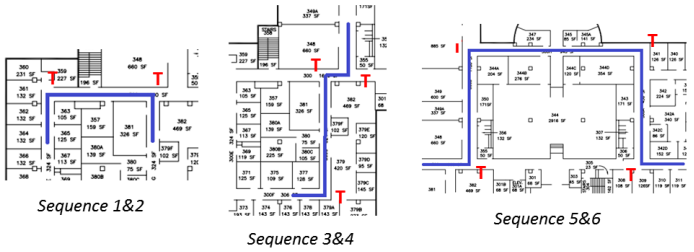

Fig. 19: Sequences performed in the turn detection test.

TABLE III: Altitude change detection test results.

Altitude Changes Detection Test	Test 1 result	Test 2 result	Test 3 result	Test 4 result	Test 5 result	Number of Errors
Sequence 1 (D)	D	D	D	D	D	0
Sequence 2 (U)	U	U	U	U	U	0
Sequence 3 (DD)	DD	DD	DD	DD	DD	0
Sequence 4 (UU)	UU	UU	UU	UUU	UU	1
Sequence 5 (DUU)	DUU	DUU	DUU	DUU	DUUU	1
Sequence 6 (DDU)	DDU	DDU	DDU	DDU	DDU	0

for a total of 30 tests and 60 altitude changes. Table III details the results of the experiments and the number of errors. As it can be seen from the table, out of 60 altitude changes in 30 experiments, the algorithm only made two mistakes for an accuracy of 96.6%.

D. Activity Recognition Results

Table IV shows the confusion matrix for the activity classification tests, in which the first column shows the real experiment performed and the first row the activity recognized. A set of 10 experiments were performed for every activity considered in the decision tree. In total 40 activity recognition tests were performed. As it can be seen from the table, only the elevator activity was misclassified; two out of ten times, it was classified as stairs for a 95% accuracy.

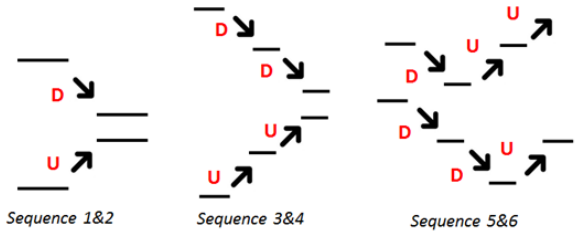

Fig. 20: Sequences performed in the altitude detection test.

TABLE IV: Activity recognition test results.

Activity Recognition Test	Stationary (Sy)	Elevator (E)	Walking (W)	Stairs (St)	Number of Errors
Stationary (Sy)	10	-	-	-	0
Elevator (E)	-	8	-	2	2
Walking (W)	-	-	10	-	0
Stairs (St)	-	-	-	10	0

E. Activity Specification Results

Once the activity is recognized, some of them require additional algorithms to enable a complete tracking of the user. In the case of the walking activity, the step detection and counting algorithm returns the number of steps walked. In the case of the stairs activity, the stair detection and counting algorithm provides with the number of stairs. The results of the evaluation of these two algorithms are analyzed in this section.

1) Step Detection and Counting Test: The step detection and counting algorithm was tested executing three walking sequences with different distance of 5, 10, and 15 steps. Each sequence was repeated 10 times, for a total of 30 walking traces and 300 steps. Table V shows the results, in which the first column is the real number of steps walked and the first row indicates the number of steps detected relative to the correct value. The content in the table shows the number of experiments for each possible number of mistakes. As it can be seen from the table, 6, 7, and 8 times out of 10 experiments in each distance, the algorithm counted the number of steps correctly and in most of the error cases, the algorithm counted one step difference only. In total, out of 300 steps, only 9 errors were detected for a 97% accuracy.

TABLE V: Steps detection and counting test results.

Step Counting Test	-2 steps	-1 step	Correct	+1 step	Number of Errors
Distance 1: 5 steps	-	3	7	-	3
Distance 2: 10 steps	-	2	8	-	2
Distance 3: 15 steps	1	2	6	1	4

2) Stair detection and counting test results: Similar to the previous test, the stair detection and counting algorithm was evaluated testing 2 stair sequences of 6 and 12 stairs, respectively. Each sequence was repeated 10 times (combining up and down direction), making a total of 20 motion traces and 180 stairs. Table VI shows the results following the same format used in the previous evaluation. Similarly, most of the steps were counted correctly and, in this case, all the mistakes were cases with a difference of plus or minus one step. From a total of 180 steps, only 6 errors were detected for a 96.7% accuracy.

TABLE VI: Stairs detection and counting test results.

Stair Counting Test	-2 stairs	-1 stair	Correct	+1 stair	Number of Errors
Motion 1: 6 stairs	-	2	8	-	2
Motion 2: 12 stairs	-	3	6	1	2

In addition, we compared our step/stair detection and counting algorithms with the most recent step detection and counting algorithm included in the last version of the Android platform (version 4.4.2). It is worth noticing that, although this solution is available through Google Play services, it is actually a hardware-based implementation, as it uses a dedicated chip included in the LG Nexus 5 smartphone for reading and analyzing the raw data coming from the accelerometer; therefore, this feature is not available in most of the current smartphones in the market.

TABLE VII: Comparison step/stair detection and counting for hardware and software solutions.

Comparison Hardware vs Software Solutions	Test 1		Test 2		Test 3		Test 4		Test 5	
	G - HW	*P - SW*	*G-HW*	*P-SW*	*G-HW*	*P-SW*	*G-HW*	*P-SW*	*G-HW*	*P-SW*
Walking: 10 steps	10	10	9	10	10	11	9	11	10	11
Walking: 20 steps	21	19	20	19	21	20	20	20	21	19
Climbing up: 13 stairs	12	12	14	12	13	13	13	13	14	13
Climbing down: 13 stairs	13	13	13	14	14	12	14	12	13	13
Combined: 6 St- 3 W- 6 St	16	15	15	14	16	14	15	15	15	14

We compared both algorithms performing five different sequences in texting position with a LG Nexus 5 smartphone. The sequences were: walking segments with distances of 10 and 20 steps; taking the stairs up and down 13 stairs; and a combined motion trace involving 6 stairs, walking 3 steps and 6 additional stairs. Each experiment was repeated 5 times for a total of 25 sequences and more than 355 steps and stairs. Table VII shows the results of the experiments in which the first column represents the real experiments performed and the two first rows are the header of the results detected by the two solutions under study. $G-HW$ means the Google hardware-based solution and $P-SW$ refers to the Proposed Software solution (our algorithms). As it can be seen from the table, both solutions present very similar and accurate results. Out of 355 steps, our algorithm made 14 mistakes (96.1% accuracy) while Google's solution made 12 mistakes (96.7% accuracy).

F. Pedestrian Tracking Results

This section presents the combined tracking accuracy when the complete architecture of the system is applied. In this evaluation, we utilized three traces, each representing a different type of motion inside a building.

The first experiment mimics a person walking inside the same floor through the corridors in a rectangle manner. It includes four walking segments and four turns, as shown in Figure 21. The walking sections are $18m$ (23 steps approximately) and $14m$ (18 steps approx.) long, respectively. Five independent tests were performed in both, clockwise (Table VIII) and counterclockwise (Table IX) directions. The tables show the results of the walking segments since the corners were detected correctly. As it can be seen from the tables, the results are very accurate.

TABLE VIII: Tracking rectangle motion clockwise results.

Rectangle Motion	W 1 (23)	W 2 (18)	W 3 (23)	W 4 (18)	Total Steps (82)	Number of Errors
Test 1	23	18	23	18	82	0
Test 2	22	19	22	19	82	4
Test 3	22	19	23	19	83	3
Test 4	23	18	23	19	83	1
Test 5	22	19	23	19	83	2
Average	22.4	18.6	22.8	18.8	82.6	2.4

The second experiment mimics a person in a sequence of walking, stairs and turns, as shown in the left sequence in Figure 22). The activity was performed in both ways, going down and up directions, five times each. The sequence, from up to down, consists of the following segments: walking $2m$ (3 steps approximately), 12 stairs, turn, walking $3m$ (4 steps approx.), turn, 7 stairs, and $2m$ walking. Table X summarizes the results for this experiment and Table XI shows the results when performing the activity in the opposite direction, i.e., from down to up. Turns were correctly detected. Again, it can be seen from the tables that the algorithms are very accurate.

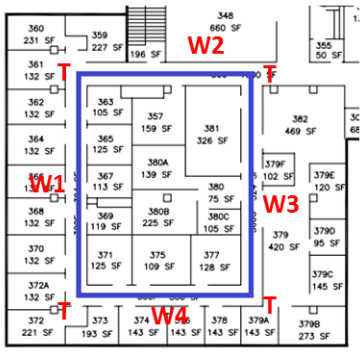

Fig. 21: Motion traces used for testing square motions.

TABLE IX: Tracking rectangle motion counterclockwise results.

Rectangle Motion	W 4 (18)	W 3 (23)	W 2 (18)	W 1 (23)	Total Steps (82)	Number of Errors
Test 1	18	23	18	24	83	1
Test 2	18	23	19	23	83	1
Test 3	18	23	17	23	81	1
Test 4	18	24	18	24	84	2
Test 5	18	23	18	24	83	1
Average	18	23.2	18	23.6	82.8	1.2

TABLE X: Results of taking the stairs down.

Tracking Down Stairs Motion	W 1 (3)	St 1 (12)	W 2 (4)	St 2 (7)	W 3 (2)	Number of Errors
Test 1	2	10	3	9	2	7
Test 2	3	10	4	8	3	3
Test 3	2	13	3	8	3	4
Test 4	3	12	2	7	3	2
Test 5	2	11	2	7	4	5
Average	2.4	11.2	2.8	7.8	3.2	4.2

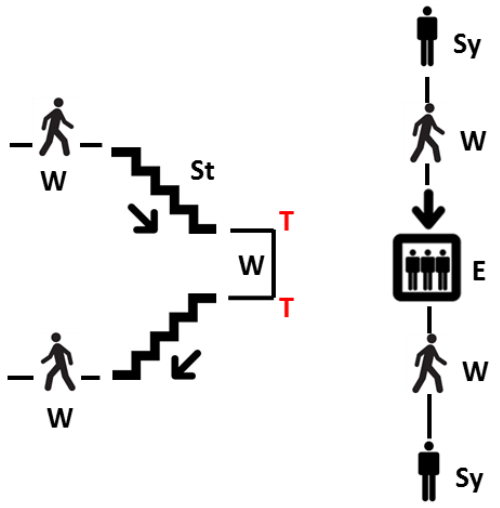

Fig. 22: Motion traces for testing pedestrian indoor tracking.

TABLE XI: Results of taking the stairs up.

Tracking Up Stairs Motion	W 3 (3)	St 2 (7)	W 2 (4)	St 1 (12)	W 1 (3)	Number of Errors
Test 1	2	7	2	12	3	3
Test 2	3	9	2	12	3	4
Test 3	4	7	4	12	4	2
Test 4	2	8	3	11	3	4
Test 5	4	7	3	11	2	4
Average	3	7.6	2.8	11.6	3	3.4

The last motion mimics a person that combines several activities in different floors: stays stationary, walks, and takes the elevator, as shown at the right hand side of Figure 22). The sequence consists of the following segments: stationary, walking $1.5m$ (2 steps approximately), turn, 2 floors in the elevator ($5sec$ approx.), and $1.5m$ walking. Table XII summarizes the results for this experiment in which tests 1 to 5 show the result of the user going in down and tests 6 to 10 going up. Turns and stationary periods were correctly detected. As the table shows, the results are also very accurate.

From these results, it can be inferred that the combination

TABLE XII: Results of combined sequence in different floors.

Combined Motion	W 1 (2 steps)	E (5 sec)	W 2 (2 steps)	Total Steps (4)	Number of Errors
Test 1	2	5	2	4	0
Test 2	2	4	2	4	1
Test 3	2	6	3	5	2
Test 4	2	5	3	5	1
Test 5	2	6	3	5	2
Test 6	2	5	2	2	0
Test 7	2	4	2	4	1
Test 8	3	5	3	6	2
Test 9	2	5	3	5	1
Test 10	2	5	2	4	0
Average	2.1	5	2.5	4.4	1

of the individual modules added complexity to the tracking problem. The separators used to segment the original trace (turns and altitude changes) can split the motion in wrong samples that lead to errors in the final activity specification. Also, the quick transition among activities in pedestrian movements, as a consequence of the granularity of the indoor environment, requires a high accuracy, not only in the classification of the activity, but also in the detection of the exact moment when the action was performed. While a more rigorous experimentation is necessary (across more buildings, users, phones, and models), the results obtained from these experiments justify the proposed multisensory indoor tracking system as a good solution. Further, the individual algorithms can be used separately in many other applications, such as in pedometers.

VI. Conclusions and Future Research

This paper presents a model to track pedestrians in indoor environments using the sensors embedded in current off-the-shelf smartphones. The model includes four modules to collect the raw sensor data, segment the data using corners, stairs, and elevators, identify the activity of the user, and quantify the type of activity. A series of algorithms and mechanisms are included in each module to achieve these objectives. For example, there are algorithms to detect corners, stairs, and elevators as well as others to accurately count the number of steps and stairs taken by the user. Similarly, the activity recognition module recognizes whether the person is stationary, moving, taking the stairs or an elevator. In addition, a complete testbed framework that simplifies the systems evaluation process is implemented.

The performance evaluation shows very good results in both, the individual algorithms and the overall system with all the modules and algorithms combined. The evaluation of the system shows a combined tracking accuracy of 91.06%. Individually, the activity recognition module recognized the activities correctly 95% of the time; the step counting and stair counting algorithms are 97% and 96.7% accurate, respectively; and the detection of user changes of direction and altitude are shown to be 98.8% and 96.6% accurate, respectively.

The proposed work can be extended and improved in several directions. First, the presented evaluation was performed using the smartphone in one position, the messaging position. A complete solution should be able to track the user regardless of the position of the smartphone, or at least using other common positions such as calling, swinging (refers to the position in which the user holds the device in a hand while walking) and pocket (the device sits in the user's pocket/bag, which is the most common position when the user is walking). Since most of the algorithms use the magnitude of the signal, they should potentially work in any of the described positions. However, this needs further investigation and evaluation. Second, the current design of the system tracks pedestrian motions offline, meaning that the sensor data are initially gathered by the mobile application and, once the motion is finished, they are sent to the server, which applies the different algorithms and returns the results. We are currently modifying the system to track pedestrian movements in real time, collecting and processing the data and returning the results directly on the smartphone. An initial implementation was utilized in the experiments where we compared our step/stair detection and

counting algorithms with those of Google's. Third, in the current version of the system, the data collection module is initiated manually by the user when entering an indoor location. A useful improvement would be an additional module responsible for changing from outdoor location services to this indoor location system and viceversa automatically. Fourth, we are also working on the integration of this system into a complete application for indoor navigation where the indoor map data is provided a priori. The current position could be displayed on the map on real time, the system could provide navigation indications to the pedestrian and also track the user movements to correct the indications if needed. Finally, in the future online navigation system, an investigation of the power consumption of the system would be recommended.

ACKNOWLEDGMENTS

This work has been partially funded by the Department of Science and Technology (Spain) under the National Program for Research, Development and Innovation (projects TIN2011-25978 entitled Obtaining Adaptable, Robust and Efficient Software by including Structural Reflection to Statically Typed Programming Languages and TIN2009-12132 entitled SHUBAI: Augmented Accessibility for Handicapped Users in Ambient Intelligence and in Urban computing environments) and by the Principality of Asturias to support the Computational Reflection research group, project code GRUPIN14-100.

REFERENCES

[1] M. Alzantot and M. Youssef. CrowdInside: automatic construction of indoor floorplans. In *proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 99–108, 2012.

[2] M. Alzantot and M. Youssef. Uptime: Ubiquitous pedestrian tracking using mobile phones. In *proceedings of IEEE Wireless Communications and Networking Conference*, pages 3204–3209, 2012.

[3] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. SurroundSense: mobile phone localization via ambience fingerprinting. In *proceedings of the 15th annual international conference on mobile computing and networking*, pages 261–272, 2009.

[4] Mortaza S. Bargh and Robert de Groote. Indoor localization based on response rate of bluetooth inquiries. In *proceedings of the First ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments*, MELT '08, pages 49–54, New York, NY, USA, 2008. ACM.

[5] Juan A. Besada, Ana M. Bernardos, Paula Tarro, and Jos R. Casar. Analysis of tracking methods for wireless indoor localization. In *proceedings of Wireless Pervasive Computing*, 2007.

[6] Y. Capelle, M. Toledo GMV, D. Kubrak, M. Monnerat, A. Mark GMV, and D. Jimnez ESA. DINGPOS: a hybrid indoor navigation platform for GPS and GALILEO. *ION GNSS*, 2008.

[7] Jaewoo Chung, Matt Donahoe, Chris Schmandt, Ig-Jae Kim, Pedram Razavai, and Micaela Wiseman. Indoor location sensing using geo-magnetism. In *proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 141–154, 2011.

[8] Ionut Constandache, Xuan Bao, Martin Azizyan, and Romit Roy Choudhury. Did you see bob?: human localization using mobile phones. In *proceedings of the sixteenth annual international conference on mobile computing and networking*, pages 149–160, 2010.

[9] Ionut Constandache, Romit Roy Choudhury, and Injong Rhee. Towards mobile phone localization without war-driving. In *proceedings of IEEE INFOCOM*, pages 1–9, 2010.

[10] Carl Fischer, Poorna Talkad Sukumar, and Mike Hazas. Tutorial: implementation of a pedestrian tracker using foot-mounted inertial sensors. *IEEE Pervasive Computing*, 2012.

[11] J.R. Gonzalez Hernandez and C.J. Bleakley. Accuracy of spread spectrum techniques for ultrasonic indoor location. In *proceedings of the 15th International Conference on Digital Signal Processing*, pages 284–287, 2007.

[12] Yanying Gu, Anthony Lo, and Ignas Niemegeers. A survey of indoor positioning systems for wireless personal networks. *IEEE Communications Surveys & Tutorials*, 11(1):13–32, 2009.

[13] Robert Harle. A survey of indoor inertial positioning systems for pedestrians. *IEEE Communications Surveys & Tutorials*, pages 1–13, 2013.

[14] V. Hovinen, M. Hamalainen, and T. Patsi. Ultra wideband indoor radio channel models: preliminary results. In *proceedings of the IEEE Conference on Ultra Wideband Systems and Technologies*, pages 75–79, 2002.

[15] A. R. Jimenez, F. Seco, C. Prieto, and J. Guevara. A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU. In *proceedings of the IEEE International Symposium on Intelligent Signal Processing*, pages 37–42, 2009.

[16] Yunye Jin, Mehul Motani, Wee-Seng Soh, and Juanjuan Zhang. Sparse-Track: enhancing indoor pedestrian tracking with sparse infrastructure support. In *proceedings of IEEE INFOCOM*, pages 1–9, 2010.

[17] F. Lassabe, P. Canalda, P. Chatonnay, and F. Spies. Indoor wi-fi positioning: techniques and systems. *Annals of Telecommunications*, 64(9-10):651–664, October 2009.

[18] Hui Liu, H. Darabi, P. Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1067–1080, 2007.

[19] Carsten Magerkurth, Adrian David Cheok, Regan L. Mandryk, and Trond Nilsen. Pervasive games: Bringing computer entertainment back to the real world. *Comput. Entertain.*, 3(3):4–4, July 2005.

[20] S. S. Manapure, H. Darabi, V. Patel, and P. Banerjee. A comparative study of radio frequency-based indoor location sensing systems. In *proceedings of the IEEE International Conference on Networking, Sensing and Control*, volume 2, pages 1265–1270 Vol.2, 2004.

[21] Martin Mladenov and Michael Mock. A step counter service for java-enabled devices using a built-in accelerometer. In *proceedings of the 1st international workshop on context-aware middleware and services*, pages 1–5, 2009.

[22] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. LANDMARC: indoor location sensing using active RFID. *Wireless Networks*, 10(6):701–710, November 2004.

[23] B. Ozdenizci, Kerem Ok, V. Coskun, and M.N. Aydin. Development of an indoor navigation system using NFC technology. In *proceedings of the Fourth International Conference on Information and Computing*, pages 11–14, 2011.

[24] Guenther Retscher, Eva Moser, Dennis Vredeveld, Dirk Heberling, and Joerg Pamp. Performance and accuracy test of a WiFi indoor positioning system. *Journal of Applied Geodesy*, 1(2), January 2007.

[25] Phillip Tom, Valrie Renaudin, Okan Yalak, and Bertrand Merminod. Indoor navigation of emergency agents. *European Journal of Navigation*, 5(3):36–45, 2007.

[26] Upkar Varshney. Pervasive healthcare and wireless health monitoring. *Mobile Networks and Applications*, 12(2-3):113–127, July 2007.

[27] Alejandra Vidal, Juan Jose Marron, and Miguel A.Labrador. Real-time pedestrian tracking in indoor environments. *6th IEEE Latin-American Conference on Communications*, November 2014.

[28] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No need to war-drive: Unsupervised indoor localization. In *proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 197–210, 2012.

[29] Oliver Woodman and Robert Harle. Pedestrian localisation for indoor environments. In *proceedings of the 10th international conference on ubiquitous computing*, pages 114–123, 2008.

[30] Oliver Woodman and Robert Harle. RF-based initialisation for inertial pedestrian tracking. In *Pervasive Computing*, pages 238–255. Springer, 2009.

[31] Junhui Zhao and Yongcai Wang. Autonomous ultrasonic indoor tracking

system. In *proceedings of the International Symposium on Parallel and Distributed Processing with Applications*, pages 532–539, 2008.