2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 24-27 September 2018, Nantes, France

A Novel Step Detection and Step Length Estimation Algorithm for Hand-held Smartphones

Nicolò Strozzi, Federico Parisi, and Gianluigi Ferrari
Internet of Things Laboratory (IoTLab), Department of Engineering and Architecture (DIA)
University of Parma
Parma, Italy
Email: {nicolo.strozzi, federico.parisi}@studenti.unipr.it, gianluigi.ferrari@unipr.it

***Abstract*—In this paper, we present an innovative inertial navigation system based on the data collected through the Inertial Measurement Unit (IMU) embedded in a commercial smartphone. We propose an innovative step detection algorithm which is independent of the holding mode, the only assumption being that the device is hand-held (i.e., the user is texting/navigating or phoning) and its movement is related to the upper body displacement during walking. We also present a new approach able to automatically calibrate the step length estimation formula according to the smartphone positioning. The developed algorithms have been validated through a test campaign in which we have evaluated the system performance considering three different smartphone models and different path lengths. The obtained results show that the maximum step detection error is always below 4% (average: 2.08%; standard deviation: 1.82%) whereas the maximum path length estimation error is below 8.1% (average: 3.6%; standard deviation: 1.81%) in all the considered cases.**

***Index Terms*—inertial navigation, smartphone, dead reckoning, gait segmentation, position independent step detection**

I. Introduction

The localization problem is a challenging research field which has been extensively investigated. The ability to locate objects and/or persons in the environment is a key feature in several applications. Navigation systems, for example, exploit the knowledge of a sequence of past and predicted positions to provide the user with useful information in order to guide him to a specific destination.

In outdoor environment, the position can be obtained through a Global Navigation Satellite System (GNSS), such as the Global Positioning System (GPS) [1] or the Galileo system [2], which are based on signals transmitted by geostationary satellites orbiting around the globe. Other systems suitable for outdoor navigation are based on the triangulation of different signals, such as the 4G mobile network-based triangulation [3]. All these technologies are afflicted by common radio signal related problems, such as shadowing (i.e., the presence of an object which block the line of sight between the transmitter and the receiver), fading (i.e., variation or attenuation of the transmission channel due to environmental factors, such as surfaces which can produce multiple paths traversed by the signal) or external noise (i.e., external signals which could, intentionally ot not, disturb the transmitted signal). In spite of the described problems, these technologies are rapidly increasing their effectiveness in outdoor scenarios, whereas they still perform poorly in indoor environments, where the external signals are obstructed by the buildings.

The limitations of the GNSS-based navigation technologies in indoor scenarios and in some problematic outdoor environments, such as forests or cities with urban canyons, have led to the adoption of alternative localization systems, based on different radio signals, such as WiFi [4], [5], Bluetooth Low Energy (BLE) [6], [7], and Ultra Wide Band (UWB) [8]. All these approaches rely on the use of transmitting devices placed in well known points to triangulate the position of the user's receiving device, and are able to achieve very good performance in indoor environments. However, the costs and complexity related to the need of a pre-deployed infrastructure and its maintenance, represent the main limitation to the extensive adoption of these technologies.

Inertial-based solutions can overtake these issues because of their independence from external infrastructures. Inertial Navigation Systems (INSs) exploit data collected through IMUs (which usually embed a three-axial accelerometer, a three-axial gyroscope, and a three-axial magnetometer) to continuously estimate the position, orientation and velocity of a moving object/user. Systems such as the ones presented in [9], [10] rely on the data collected from IMUs placed on the feet or on the upper trunk to estimate the test subject's traveled path. As shown in [11], however, IMU placement on the body highly influences the algorithms' performance. In particular, if the sensor movement mirrors the body segment's movement (i.e., the IMU is rigidly bound to the user's foot or trunk), it is possible to take advantage of the walking dynamics to accurately estimate the horizontal displacement. A complete overview on the differences between multiple INSs is presented in [12]. Given the massive diffusion of Micro Electro-Mechanical Systems (MEMS) IMUs in commercial devices such as smartphones, a further challenge for INSs consists in using data collected from the IMU embedded in smartphones to enable low-cost and infrastructure-free personal navigation, as the smartphone-based map-aided INS presented in [13].

In this paper, we propose a novel approach for pedestrian navigation based on inertial data collected from hand-held smartphones. In Section II, we first provide an in-depth description of the features of the Android operating system on which our algorithm is run; then, we introduce our innovative

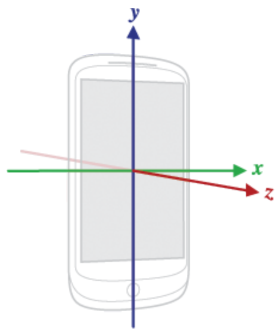


Fig. 1. Android smartphone reference system.

strategies to perform a smartphone-based step detection and step length estimation. In Section III, the system's performance, considering multiple hardware platforms and different experimental set-ups, is evaluated. The details of the developed Matlab scripts and custom-built Android application are also presented. Finally, in Section IV conclusions are drawn.

II. Methods

A. Android Sensors APIs

The algorithms described in the following rely on data collected from Android smartphones. The Android operating system provides a set of APIs for managing, independently from the hardware, the raw data streams from the smartphone sensors. The complete APIs documentation can be found at the Android developer website [14]. The mechanism used to obtain data from the sensors consists in creating and activating a listener (i.e., `SensorManager.registerListener`) over the required sensors, which throws a callback every time the data are available. It is important to remark that a sensor, in this context, could be both an actual physical sensor or an emulated one, which means that the Android operating system retrieves the data from the hardware components whenever available or derives some measurements combining data from the available physical sensors (e.g., by combining the acceleration, the angular velocity, and the magnetic field it is possible to obtain the orientation as if it had been measured from a hardware orientation sensor). However, the list of available sensors depends on the specific smartphone, i.e., different smartphones may likely embed different sensors. Therefore, in order to enable the listener over a specific sensor, it is necessary to call the following function:

```
SensorManager.registerListener(this,
SensorManager.getDefaultSensor(SENS_TYPE),
SAMPLE_RATE);
```

where: `SensorManager` is the object which manages the sensor in Android; `SENS_TYPE` is the sensor type identification code; and `SAMPLE_RATE` is the sampling rate characterized as *fastest*, *normal*, *game* (i.e., rate suitable for games), and *ui* (i.e., suitable for the user interface).

In order to perform an effective purely inertial navigation, one needs to know the orientation with respect to the geomagnetic north: this is provided by the magnetic sensor (i.e., `Sensor.TYPE_MAGNETIC_FIELD`) in terms of μT. The smartphone reference system is shown in Fig. 1. The orientation in the world frame can be obtained through the `Sensor.TYPE_ORIENTATION` in terms if rotation angle (in degrees) with respect to the geomagnetic north. Given that this sensor has been deprecated in from API level 8 (i.e., Android 2.2 Froyo), in the latest Android version[1] it is possible (i) to exploit the `getRotationMatrix()` function which use the gravity sensor (`Sensor.TYPE_GRAVITY`) and the magnetic sensor or (ii) to use the `Sensor.TYPE_ROTATION_VECTOR`, which is a "software sensor" which directly computes the smartphone orientation $\theta^{(w)}$ (where w represents the world reference system) as a unit vector, namely $< x, y, z >$, and an angle of revolution around that vector, namely θ. In other words, the orientation is represented by three values expressed as follows:

$$\theta_x^{(w)} = x \cdot \sin\left(\frac{\theta}{2}\right) \tag{1}$$

$$\theta_y^{(w)} = y \cdot \sin\left(\frac{\theta}{2}\right) \tag{2}$$

$$\theta_z^{(w)} = z \cdot \sin\left(\frac{\theta}{2}\right) \tag{3}$$

where: x is the axis tangential to the ground at the device's current location and pointing towards East; y is the axis tangential to the ground at the device's current location and roughly pointing towards the magnetic north; and z is the axis which points towards the sky and is perpendicular to the ground according to the axis-angle notation.

B. Obtaining Smartphone Linear Acceleration

In order to obtain the smartphone linear acceleration in the world reference system w, we collect the raw acceleration from the hardware sensor $\{acc^{(s)}(k)\}$, where s refers to the smartphone reference system in Fig. 1 and k is the sampling epoch. Therefore, we use the orientation expressed in angle-axis notation $\theta^{(w)}(k)$ obtained through the rotation vector software sensor to estimate the rotation matrix $R^{(s,w)}(k)$. First, we obtain the corresponding quaternion as follows

$$q_0' = 1 - \left(\theta_x^{(w)}\right)^2 - \left(\theta_y^{(w)}\right)^2 - \left(\theta_z^{(w)}\right)^2 \tag{4}$$

$$q_0 = \begin{cases} \sqrt{q_0'} & \text{if } q_0' > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$q_1 = \theta_x^{(w)} \tag{6}$$

$$q_2 = \theta_y^{(w)} \tag{7}$$

$$q_3 = \theta_z^{(w)} \tag{8}$$

where $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ is the orientation expressed in quaternion form (q_0, q_1, q_2, q_3 are real numbers and $\mathbf{i}, \mathbf{j}, \mathbf{k}$

[1]In this study, the Android APIs level 23 (i.e., Android Marshmallow 6.0) have been used.

are the fundamental quaternion imaginary units). The rotation matrix at epoch k can then be obtained as follows:

$$R^{(s,w)}(k) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix}. \quad (9)$$

This matrix is used to rotate the raw acceleration according to the following equation:

$$acc^{(w)}(k) = acc^{(s)}(k) \cdot R^{(s,w)}(k) \quad (10)$$

moving to the world reference system obtaining $acc^{(w)}(k)$. The linear acceleration $acc^{(w)}_{\text{lin}}(k)$ is then obtained by subtracting the gravity $g = 9.81\ \text{m/s}^2$ to the rotated acceleration z-axis component $acc^{(w)}_z(k)$, i.e., $acc^{(w)}_{\text{lin}}(k) = acc^{(w)}(k) - [0\ 0\ g]$.

C. Step Detection

The following approach allows us to check the gait status every sample and, thus, to detect a step in near-real time. In fact, the idea behind the proposed step detection algorithm relies on the assumption that, when we are close to a Heel-Strike (HS) or a Toe-Off (TO), the principal component in the acceleration magnitude is the vertical one. Therefore, if the vertical acceleration is "close enough" to the acceleration magnitude, the step is detected. The acceleration magnitude at the k-th sample can be computed as follows:

$$a^{(m)}(k) = \sqrt{acc^{(w)}_{\text{lin},x}(k)^2 + acc^{(w)}_{\text{lin},y}(k)^2 + acc^{(w)}_{\text{lin},z}(k)^2} \quad (11)$$

where $acc^{(w)}_{\text{lin},x}(k)$, $acc^{(w)}_{\text{lin},y}(k)$, and $acc^{(w)}_{\text{lin},z}(k)$ are the linear acceleration components.

A state transition diagram-based approach is applied in order to describe the detection phases. A state transition happens in correspondence to every sample. In particular, the following three states are introduced.

- **No step:** this is the idle status, in which no step is detected. This happens if the test subject is static or if the previous step is ended and the next one has still to be detected.
- **Step initial phase:** in this status, the algorithm has detected a step. The smartphone is swinging because of the walking dynamics and the test subject foot is performing a HS. In this status, the vertical acceleration is maximum.
- **Step terminal phase:** this state lasts until the end of a step is detected, namely the vertical acceleration becomes higher than the magnitude threshold th_m. The horizontal displacement is then computed and the end of a cycle is declared. The system will return in the "No step" status.

As visible in the state transition diagram in Fig. 2, we define three indicator functions which allow to discriminate state transitions (i.e., to distinguish the gait phases). The first condition, namely α_1, is defined as follows

$$\alpha_1(k) = \begin{cases} 1 \text{ if} \begin{cases} a^{(m)}(k) & > th_m \\ |a^{(m)}(k) - acc^{(w)}_{\text{lin},z}(k)| & < th_d \\ t(k) - t(LSI) & < min_d \end{cases} \\ 0 \text{ otherwise.} \end{cases} \quad (12)$$

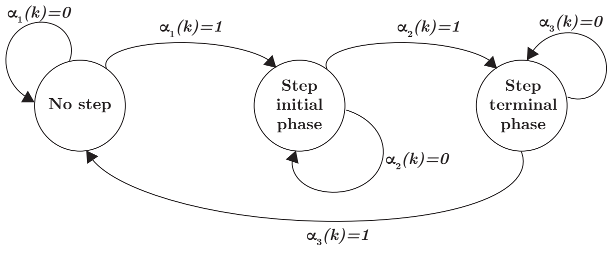


Fig. 2. Step detection algorithm state diagram.

where: th_m is the magnitude threshold (dimension: [m/s^2]), th_d is the "vertical-to-magnitude-similarity" threshold, namely the distance between the acceleration magnitude and the vertical acceleration which enables the step detection (dimension: [m/s^2]); min_d is the minimum distance, in terms of number of samples, between a step and next one; and the Last Step Index (LSI) is the sample index corresponding to the maximum vertical acceleration during the last step. If a step has not been detected yet, LSI is initialized to 0. The chosen values for the considered thresholds are: $th_m = 1.5$ m/s^2; $th_d = 0.5$ m/s^2; and $min_d = 50$ samples, respectively.

If $\alpha_1(k) = 1$, the status is updated to "Step initial phase" and the algorithm checks a second variable α_2 in order to detect the end of the first part of a step (i.e., the HS of one foot which approximately corresponds to the TO of the other foot). The second condition is described through the following indicator function:

$$\alpha_2(k) = \begin{cases} 1 & \text{if} \quad acc^{(w)}_{\text{lin},z}(k) < 0 \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

The use of $\alpha_2(k)$ allows to detect the moment in which the decreasing vertical acceleration crosses zero. This means that the HS event has occurred and the current step's acceleration maximum can be identified as follows:

$$a_{\max}(i) = \max_{k \in \text{step}} acc^{(w)}_{\text{lin},z}(k) \quad (14)$$

where i represents the step index. The index LSI is then updated as the epoch of the last acceleration maximum: therefore, in every step, LSI is the epoch associated with the maximum $a_{\max}(i)$ of the current step.

The algorithm status is then updated in order to detect the epoch i corresponding to the minimum acceleration $a_{\min}$ in the current step. This condition is defined through the following indicator function:

$$\alpha_3(k) = \begin{cases} 1 & \text{if} \quad acc^{(w)}_{\text{lin},z}(k) > th_m \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

Once the third condition is satisfied (i.e., $\alpha_3 = 1$), it is possible to identify the minimum in the vertical acceleration in order to estimate the horizontal displacement through one of the various algorithms which rely on the detection of acceleration peaks and valleys [15], [16]. In our approach, we use the Weinberg *et al.* estimation algorithm [17], as will be described in more detail in Subsection II-D.

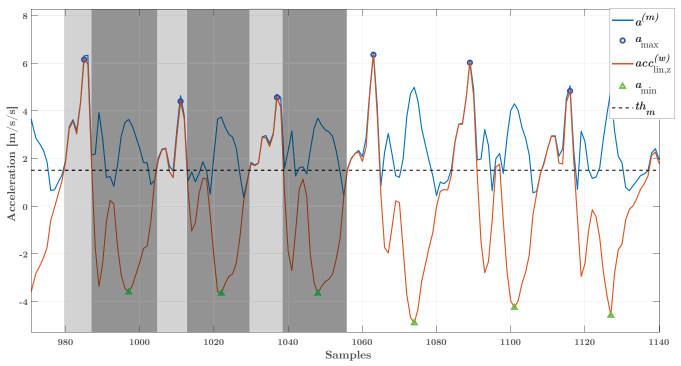


Fig. 3. Gait segmentation obtained through the proposed algorithm with smartphone held in texting mode. The blue line represents the acceleration magnitude $a^{(m)}$, the red line is the vertical acceleration $acc^{(w)}_{\text{lin},z}$. The light gray highlighted graph portion represents the "Step initial phase", whereas the dark Grey highlighted graph portion represents the "Step terminal phase".

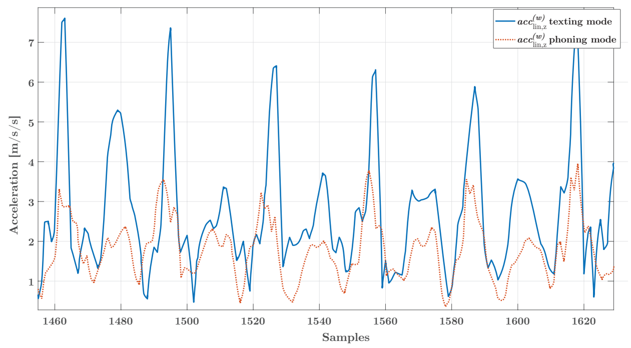


Fig. 4. Comparison between the acceleration magnitude calculated according with equation (11) for the smartphone held in texting/navigation mode (blue line) and in phoning mode (red dotted line). The two signals refer to two different trials carried out with the same smartphone.

The gait segmentation obtained through the proposed method is shown in Fig. 3, in which one can observe the signal collected from the smartphone held in navigation/texting mode. The vertical acceleration represents the main component in the acceleration magnitude during a normal walk pattern and this allows to perform step detection. Moreover, this method allows to distinguish between the acceleration peaks generated by the step with respect to the other spurious peaks in the vertical acceleration.

D. Step Length Estimation and Path Calculation

Once the gait phases are segmented as described in Subsection II-C, the Weinberg algorithm is applied and the length of the i-th step can be estimated as follows:

$$SL(i) = K \cdot \sqrt[4]{a_{\max}(i) - a_{\min}(i)} \tag{16}$$

where: K is a constant which rely on the smartphone positioning and the subject height; i is the step index; and $a_{\max}(i)$ and $a_{\min}(i)$ are the maximum and the minimum acceleration during the i-th step. The constant K is a key value in the path estimation, especially for the proposed algorithm which aims to be independent of the smartphone position. In Subsection II-E an innovative K estimation approach will be proposed.

Once the step has been detected and the step length has been computed, the position estimate $p(i) = (p_x(i), p_y(i))$ is obtained by projecting the step length along the heading direction $\psi(i)$ derived from the quaternion $q(i)$ (see Subsection II-B) as follows:

$$\psi(i) = \arctan 2(\,2\,(q_0q_3 + q_1q_2),\, 1 - 2\,(q_2^2 + q_3^2)\,). \tag{17}$$

The new position is then estimated as follows

$$\begin{aligned} p_x(i) &= SL(i) \cdot \cos(\psi(i)) + p_x(i-1) \\ p_y(i) &= SL(i) \cdot \sin(\psi(i)) + p_y(i-1). \end{aligned} \tag{18}$$

E. Estimation of K

Different smartphone placements are related to different acceleration magnitudes during walking. In fact, the way the user is holding the smartphone determines its orientation and its potential movements. When the smartphone is held in phoning mode, for example, it is subject to a vertical fasten force stronger compared to the one experienced in the texting/navigation holding mode, which is where the hand is free to move along the vertical direction. This different acceleration magnitude behaviors are illustrated in Fig. 4. The Weinberg step length estimation formula shown in equation (16) relates the trunk vertical acceleration to the horizontal displacement by assuming the sensor attached to the upper body. This assumption no longer holds when the sensor (i.e., the embedded IMU in the smartphone) is hand held. In this case, the formula has to be calibrated not only according to the test subject walking characteristic, but also according to the sensor degrees of freedom.

The idea behind our approach is to adopt a constant K for the step length estimation which relies on the acceleration magnitude as "motion quantity indicator." Our experimental tests have lead to the development of the following novel formula which allows us to automatically calibrate the step length estimator according to the smartphone position:

$$K = \beta \cdot \frac{1}{\sqrt[3]{a^{(m)}_{\max}(i)}} \tag{19}$$

where: β is a per-subject estimated constant; $a^{(m)}_{\max}(i)$ is the maximum value in the acceleration magnitude during the considered i-th step. Experimentally, we found that, for the test subject whom the following results refer to, $\beta = 0.7$ is an efficient choice.

In the following section, the preliminary results obtained considering multiple experimental set-ups with a single test subject are presented.

III. Performance Evaluation

The experiments have been carried out in the Department of Engineering and Architecture of the University of Parma (latitude: 44.765277, longitude: 10.308277), along the main corridor, where markers (i.e., paper tags attached to the corridor's walls) were placed every 5 m. In order to obtain accurate

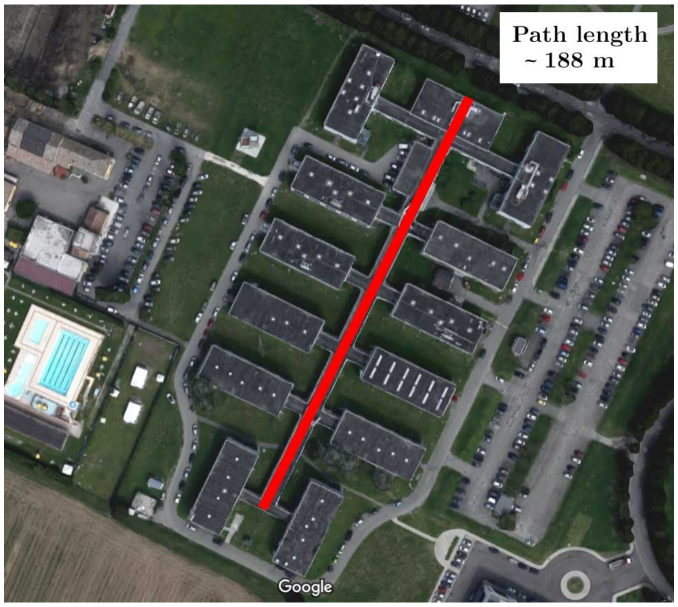


Fig. 5. Department of Engineering and Architecture of the University of Parma. The trials have been carried out in indoor environment along the main corridor (highlighted in red) which has approximate length of 188 m.

measurements, the final distance (i.e., the one between the final marker and the ending position) has been measured with a measuring tape. Every test have been video recorded to obtain (as a reference) the precise step count. As shown in Fig. 5, the corridor length is approximately 188 m: for the longest tests (i.e., the walks with 500 steps), the subject has performed a single curve with no step, namely by turning in a single movement. In order to evaluate the step detection accuracy together with the step length estimation capability, several tests have been performed. The data have been analyzed through a Matlab-based script. An Android application has been developed in order to test the navigation algorithm in real-time.

A. Hardware and Software

The data has been collected through three different smartphones, in order to verify the effectiveness of our algorithm over different hardware platforms. The chosen smartphones are (i) a Samsung S7, (ii) a Nokia 6, and (iii) a OnePlus 5T. The smartphones technical specifications are shown in Table I. The

TABLE I
ANDROID SMARTPHONES

Producer	Model	Android version	Accelerometer	Sensor manufacturer
Samsung	S7	Nogaut 7.0	K6DS3TR	STM
Nokia	6	Oreo 8.0	BMA2X2	BOSCH
OnePlus	5T	Oreo 8.0	BMI160	BOSCH

offline algorithm verification and the performance evaluation have been carried out with Matlab by analyzing the inertial data stored in different CSV files in the internal memory of

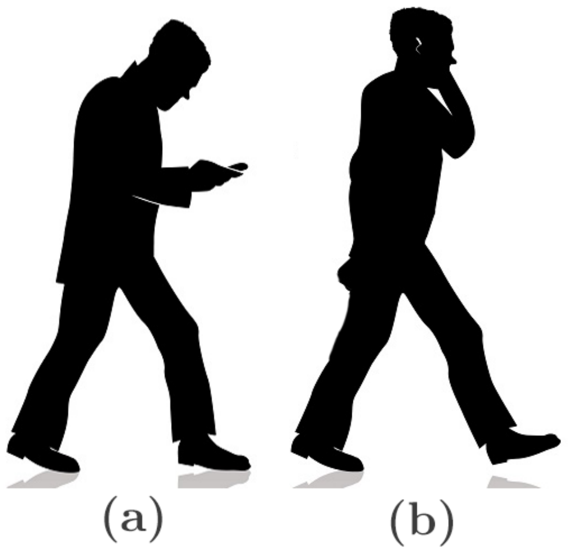


Fig. 6. The smartphone holding mode tested in the present research work. Mode (a): the smartphone is held in front of the user with one hand, allowing the direct control of the current estimated path. Mode (b): the smartphone is held as in a normal phone call, with the smartphone close to the user's ear.

the smartphone, using a third-party application for the data collection. The real-time algorithm evaluation, instead, has been performed through an ad-hoc application developed in Android Studio, which has been installed over the three smartphones. The application details are illustrated in Subsection III-D

B. Smartphone Positioning

Straightforward approaches to perform smartphone-based inertial navigation usually involve a mode recognition phase, which has to be performed before starting the step detection procedure, in order to tune the algorithm according to the specific smartphone positioning. Our approach is *mode-independent*, which means that we only assume that the smartphone is placed in the upper part of the body (i.e., in the user's hand in texting/navigation mode or next to the head on phoning mode). The proposed algorithm, in fact, is based on the upper body acceleration dynamics: in particular, it compares the vertical acceleration and the acceleration magnitude to perform the step detection. The vertical acceleration is computed online and refers to the world frame coordinates, enabling the step detection independently from the current smartphone orientation.

In order to check the system accuracy through experimental tests, we have chosen two main smartphone use modes, which are usually involved in the normal walk: the texting/navigation mode and the phoning mode, as shown in Fig. 6. It is important to remark that there is no difference, in terms of system performance, between the cases where the smartphone is held with the right or the left hand.

C. Results

In Table II and Table III, the results obtained in the experimental trials are summarized. The experimental data have been organized according with the smartphone use modes.

TABLE II
SYSTEM PERFORMANCE: SMARTPHONE IN TEXTING/NAVIGATION MODE

# Step	Detected step			True dist. [m]			Estimated dist. [m]			% ϵ on dist.		
	S7	N6	5T	S7	N6	5T	S7	N6	5T	S7	N6	5T
20	20	21	20	13.9	13.5	12.3	13.3	12.4	12.8	4.3	8.1	4.6
50	49	52	52	34.5	33.1	33.5	32.9	31.7	35.9	4.6	4.1	7.1
100	101	104	96	69.7	65.7	61.4	68.2	67.9	63.2	2.1	3.3	2.9
500	503	505	491	348.5	352.2	361.2	340.7	370.5	350.3	2.3	5.2	3.2

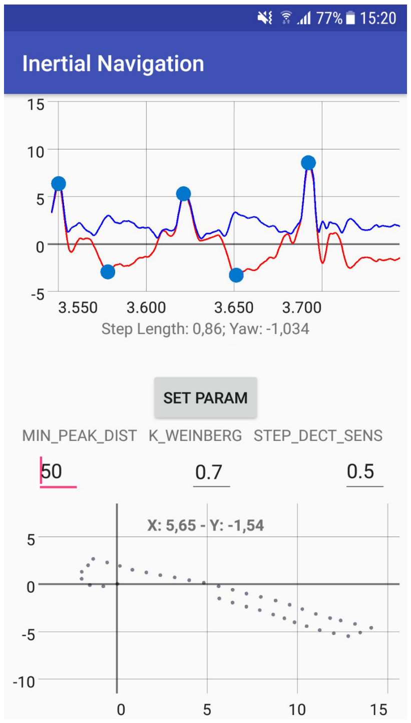
TABLE III
SYSTEM PERFORMANCE: SMARTPHONE IN PHONING MODE

# Step	Detected step			True dist. [m]			Estimated dist. [m]			% ϵ on dist.		
	S7	N6	5T	S7	N6	5T	S7	N6	5T	S7	N6	5T
20	20	21	21	14.2	13.5	13.6	14.9	14.1	13.8	4.9	4.3	1.7
50	50	49	48	35.4	33.7	33.8	34.9	35.3	33.8	1.4	4.7	0
100	101	97	101	69.1	66.7	66.5	70.4	69.4	71.2	1.8	4	1.6
500	502	498	504	351.3	345.3	352.6	365.3	332.5	344.8	3.9	3.7	2.8

In Table II, it is possible to see the algorithm effectiveness using the smartphones in texting/navigation mode. As one can see, the proposed approach can correctly detect almost all the steps, with an error, with respect to the total number of steps, which is always below 4%. These results are valid for all the three smartphones tested, i.e., the accuracy is independent of the hardware platform.

In Table III, the performance results with the smartphone in phoning mode are shown: one can observe the algorithm capability to accurately identify the steps independently of the smartphone position. This is due to the linear acceleration estimation procedure presented in Subsection II-B which allows to estimate the vertical orientation with respect to the Earth reference system. In this experimental test, as already highlighted for the previous results in Table II, the proposed approach has been able to identify the steps with high accuracy, namely, with a relative error below 3%. In Table II and Table III, it is possible to observe that the estimated distance error varies when the step number varies. This is due to error averaging over a path: more precisely, a single step length estimation error has a higher impact in shorter paths than in longer ones.

Considering the performance in terms of path length estimation, the algorithm presents a significant variance in the estimated distances, due to a high vertical displacement variability of the smartphone. Nevertheless, as can be observed in Table II and Table III, it can automatically adapt the step length estimation to the specific smartphone mode under use. In fact, the errors are comparable in both tables, which means that the bias is introduced by the common walking pattern dynamics, but not from the step length estimation formula, which is thus independent of the smartphone placement. The average (relative) error in Table II is approximately 4.3% of the total traveled distance, whereas in Table III the average (relative) error is below 3%. In Table III, it can also be observed that the results are comparable between the three smartphones, which prove that the algorithm performance is independent of the hardware, thus making the proposed approach applicable to multiple devices.



Step Length: 0,86; Yaw: -1,034

MIN_PEAK_DIST	K_WEINBERG	STEP_DECT_SENS
50	0.7	0.5

Fig. 7. A screen capture of the Android Application. The top graph represents the acceleration magnitude $a^{(m)}$ in blue, and the vertical acceleration $acc_z^{(w)}$. The light blue dots represents the maximum $a_{\max}$ and the minimum $a_{\min}$ in the vertical acceleration. The bottom graph represents the estimated path. The x and y axis are stated in meters.

It is important to remark that no modification in the step length estimation algorithm is needed to adapt the Weinberg formula to the smartphone use mode. In fact, the novel strategy introduced in equation (19), based on the acceleration magnitude, enables the automatic correction of the constant K depending on the "oscillation quantity."

D. Android Application

The Android application has been developed in order to verify the algorithm capability to reconstruct the traveled path in real-time and to provide a continuous feedback to the user. In the application layout, shown in Fig. 7, one can see (top) the acceleration (linear vertical and magnitude) with step detection/gait segmentation and, (bottom) the estimated path. The top graph is updated every time a new sample is collected from the sensor. As soon as the algorithm detects the maximum $a_{\max}(i)$ or the minimum $a_{\min}(i)$ in the vertical acceleration during the i-th step, a blue dot is displayed at the corresponding coordinates in the graph. The Android operating system can not ensure a constant sampling rate, so the proposed algorithm sets the sampling rate to the maximum possible rate (i.e., `SensorManager.SENSOR_DELAY_FASTEST`) which corresponds, for the considered smartphones, approximately to $150 \div 200$ Hz. In Fig. 7, right under the acceleration graph (on top) we also display the current step length and the current yaw (heading), which are updated every time a "*Step terminal phase*" is completed. Moreover, the application allows to set the algorithm key parameters such as: the minimum distance between two subsequent peaks, denoted as min_d (`MIN_PEAK_DIST`); the constant β (`K_WEINBERG`) needed to evaluate K according to equation (19); and the difference threshold between the acceleration magnitude and the vertical acceleration, denoted as th_d (`STEP_DECT_SENS`). In the bottom graph in Fig. 7, we also explicitly indicate the current coordinates, with respect to the starting point, expressed in meters — the default starting point is the axis origin $p(0) = (0, 0)$. In the path shown in the bottom graph, instead, every point represents a step.

IV. Conclusions

In this paper, we have presented an innovative algorithm able to estimate the traveled path of a person during walking, exploiting the data collected from the IMU embedded in commercial smartphones. We have developed two different softwares: (i) a Matlab script for offline data analysis, in order to verify the algorithm accuracy; and (ii) an Android application able to provide an online processing of the collected data, allowing real-time path estimation directly on the device screen. We have tested the algorithm performance on the basis of several trials carried out using three different commercial smartphones (namely, a Samsung S7, a Nokia 6, and a Oneplus 5T). The obtained results prove the system feasibility, in terms of accuracy and latency, with respect to the position estimation.

In our future work, we aim at validating the developed algorithm considering a larger and heterogeneous set of testing subjects, also comparing our system with smartphone-based existing solutions. We also aim to extend the set of experimental trials by analyzing different paths (i.e., stairs, curves, etc.) in order to evaluate the system performance under different working conditions. Moreover, our goal is to develop a complete Android application, which embeds indoor and outdoor maps, in order to give a feedback on the device screen about the current position in real environments. Finally, we aim at integrating other localization technologies, such as BLE-based or WiFi-based localization and GPS, in order to reduce the drift introduced from the purely inertial system and to ensure a reliable long term navigation in every scenario.

References

[1] P. Misra and P. Enge, *Global Positioning System: signals, measurements and performance second edition*. Massachusetts: Ganga-Jamuna Press, 2006.

[2] J. Benedicto, S. E. Dinwiddy, G. Gatti, R. Lucas, and M. Lugert, "Galileo: satellite system design and technology developments, european space agency," 2000.

[3] A. Ray, S. Deb, and P. Monogioudis, "Localization of lte measurement records with missing information," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, April 2016, pp. 1–9.

[4] N. Hernández, M. Ocaña, J. M. Alonso, and E. Kim, "Wifi-based indoor localization and tracking of a moving device," in *2014 Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*, Corpus Christ, TX, 2014, pp. 281–289.

[5] A. Zayets and E. Steinbach, "Robust wifi-based indoor localization using multipath component analysis," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sapporo, JP, Sept 2017, pp. 1–8.

[6] P. Barsocchi, A. Crivello, M. Girolami, F. Mavilia, and F. Palumbo, "Occupancy detection by multi-power bluetooth low energy beaconing," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sapporo, JP, 2017, pp. 1–6.

[7] P. Dickinson, G. Cielniak, O. Szymanezyk, and M. Mannion, "Indoor positioning of shoppers using a network of bluetooth low energy beacons," in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Alcala de Henares, ES, Oct 2016, pp. 1–8.

[8] M. Segura, V. Mut, and C. Sisterna, "Ultra wideband indoor navigation system," *IET Radar, Sonar Navigation*, vol. 6, no. 5, pp. 402–411, June 2012.

[9] N. Strozzi, F. Parisi, and G. Ferrari, "On Single Sensor-Based Inertial Navigation," in *IEEE EMBS 13th Annual International Body Sensor Networks Conference* (BSN 2016), San Francisco, CA, 2016.

[10] ——, "A multifloor hybrid inertial/barometric navigation system," in *IEEE International Conference on Indoor Positioning and Indoor Navigation* (IPIN 2016), Alcala de Henares, ES, Oct. 2016.

[11] ——, "Impact of on-body imu placement on inertial navigation," *IET Wireless Sensor Systems*, vol. 8, no. 1, pp. 3–9, 2018.

[12] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1281–1293, Third 2013.

[13] P. Pombinho, A. Afonso, and M. Carmo, "Point of interest awareness using indoor positioning with a mobile phone," in *211 International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS)*, Vilamoura, Portugal, March 2011, pp. 5–14.

[14] Sensors overview - android developers. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview

[15] S. Shin, C. Park, J. Kim, H. Hong, and J. Lee, "Adaptive Step Length Estimation Algorithm Using Low-Cost MEMS Inertial Sensors," in *Sensors Applications Symposium (SAS '07), IEEE*, San Diego, CA, Feb. 2007, pp. 1–5.

[16] D. Alvarez, R. Gonzalez, A. Lopez, and J. Alvarez, "Comparison of Step Length Estimators from Weareable Accelerometer Devices," in *Engineering in Medicine and Biology Society (EMBS '06), IEEE*, New York, NY, Aug. 2006, pp. 5964–5967.

[17] H. Weinberg, "Using the adxl202 in pedometer and personal navigation applications," Analog Devices, Tech. Rep. application note AN-602, 2002.