Pedestrian Dead Reconing (PDR) Algorithm for Indoor Localization

Ahmad Abdel-Qader

Electrical Engineering Department

The University of Jordan

Amman, Jordan

ahamd.m.abdelqader@gmail.com

ABSTRACT

Indoor localization has recently proven to be essential mainly because Global Positioning System (GPS) cannot provide accurate location information in the indoor environment. Also, the potential wide range of services it can provide by leveraging Internet of Things (IoT) increased the interest in it. Pedestrian Dead Reckoning (PDR) is an indoor localization method that relies on smartphone carrying sensors such as (accelerometer, gyroscope, and magnetometer). However, the inertial sensors suffer from bias and other random errors. These errors are due to misalignment of the axes in the manufacturing of the sensor or during assembly of the IMU, and due to self-heating effects.

In this report, we present a stride length estimation method, a step detection algorithm, and a sensor calibration fusion method to reduce and compensate the effects of these errors. The presented method combining outputs acquired by low-cost inertial measurement units and electronic magnetic compasses to provide the optimal three-dimensional (3D) navigation system.

TABLE OF CONTENTS

July 31, 2020

Chapter 1 Introduction

In recent years, low-cost inertial measurement units (IMUs) and magnetometer sensors are produced massively and available at a low cost. Also, because of its small size, it is widely used in mobile devices. Using mobile devices in determining the pedestrian location in an indoor environment has been a fundamental requirement in many public service applications such as navigation. It uses low-cost microelectromechanical systems (MEMS) sensors, but the raw data acquired by the sensors cannot be used directly in mathematical formulas [1].

In the presented method we implement a calibration method to calibrate the smartphone electronic compass for hard-iron and soft-iron effects. Also, implement a method to remove constant bias in the gyroscope. Using an accurate method to detect steps and calculate stride length which considered an essential part of the pedestrian dead reckoning (PDR).

Besides, using a Kalman filter in two parts: (1) combining accelerometer and gyroscope outputs to ensuring updated attitude information.

(2) combining the magnetometer and gyroscope outputs to correct the horizontal orientation.

To Be continue …

Chapter 2 Methods

The app was developed using the Android software development kit on the Android Studio Integrated development environment and tested on the Samsung A30 smartphone.

The process of building a full-function navigation app was split into 3 parts. In the first part, a step detection algorithm was implemented and tested. In the second part, an automated stride length estimation algorithm was tested and implemented. During the first two-parts, a standalone app was developed to be later incorporated into the main app. In the third part, an orientation estimation algorithm was developed, using multiple filtering stages including 2-steps Kalman filtering.

2.1 Step Detection And Stride Length Estimation Algorithm

The step detection algorithm used in this app is based on the paper [2]. To detect steps the first requirement is the smartphone linear acceleration in the world reference system. After collection the raw data from the hardware ACC its need to be referred to the world reference system by multiplying the raw acceleration with the rotation matrix R. To estimate the R we need to find the quaternion as follows:

$$q_0' = 1 - \theta_x^2 - \theta_y^2 - \theta_z^2$$

$$q_0 = \begin{bmatrix} \sqrt{q_0'} & if \ q_0' > 0 \\ 0 & otherwise \end{bmatrix}$$

$$q_1 = \theta_x$$

$$q_2 = \theta_y$$

$$q_3 = \theta_z$$

Where q is the orientation expressed in quaternion form and $\theta_x \ \theta_y \ \theta_z$ are the revolution around the axis $< x, y, z >$.

Then the rotation matrix can be found as follows:

$$R = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 1 - 2(q_1 + q_3) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix}$$

After finding R the linear acceleration ($ACCL$) can be found using the following equation:

$$ACCL = ACC \cdot R$$

Then to obtain the world reference, the linear acceleration $ACCL$ subtracted from the gravity:

$$ACCL = ACCL - [0\ 0\ g]$$

The step detection algorithm relies on the principle that when we are close to heel-strike or toe-off the vertical acceleration which as seen in Figure 1 the acceleration in the Z direction is the principal component. Therefore, when a step is detected the vertical component of the acceleration should be close enough to the acceleration magnitude ALN.
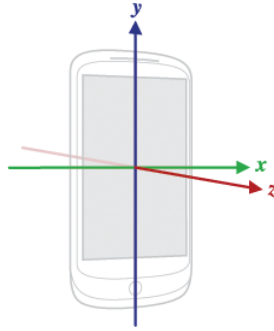

Figure 1: Phone orientation

The linear acceleration magnitude ALN can be found as follows:

$$ALN = \sqrt{ACCL_x^2 + ACCL_y^2 + ACCL_z^2}$$

The step detection algorithm is seen in Figure 2, it relays on three main states: no steps, initial phase, and terminal phase. No steps phase mean that the app doesn't record any movement and the object is static or the previous step was ended and object still in idle state. In the initial phase, in this phase, the app detects a step and the object is performing a heel-strike, the vertical

acceleration is maximum at this stage. The Terminal stage lasts until the end of the step detection process where the vertical acceleration will be more than the magnitude threshold *thm*.

The algorithm depends on three indicators which are: the *thm* is the magnitude threshold, *thd* is the "vertical-to-magnitude-similarity" threshold, namely the distance between the acceleration magnitude and the vertical acceleration which enables the step detection, *mind* is the minimum distance, in terms of the number of samples, between a step and next one; and the maximum vertical acceleration epoch number during last step *EMAX*. If a step has not been detected yet, *EMAX* is initialized to 0. The chosen values for the considered thresholds are: *thm* = 1.5 m/s2, *thd* = 0.5 m/s2, and *mind* = 50 samples, respectively.

The stride length estimation depends on the maximum vertical acceleration *AMAX* that recorded in the initial phase. Also, it depends on the minimum vertical acceleration *AMIN* which recorded in the terminal phase.

The Weinberg algorithm is applied to calculate the stride length. The equation is shown below:

$$SL = C * \sqrt[4]{AMAX - AMIN}$$

Where *SL* is the stride length in meter, *C* is a constant which rely on the smartphone positioning

and the subject height. The details on the constant *C* can be found in Subsection II-E in the paper [2].

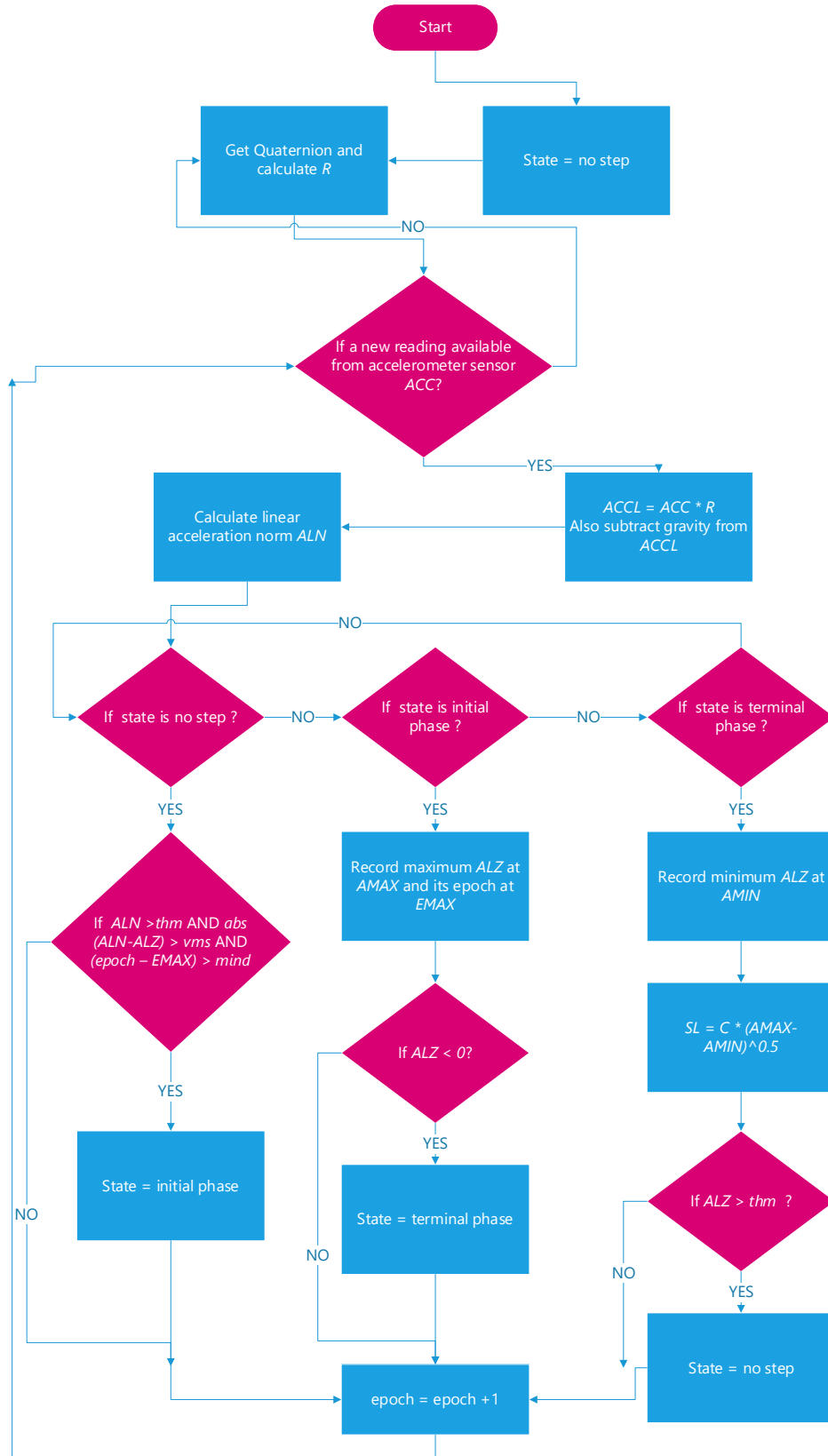The below flowchart describes the full process of the step detection and stride length estimation process.

Figure 2: Step Detection Algorithm

2.2 Calibration of Sensors

2.2.1 Gyroscope Calibration

The calibration process for the gyroscope starts by putting the mobile phone on a flat surface for a short period (until the App record enough readings). After recording enough data (in our trail we use 600 readings) the moving average for the gyroscope data is calculated to observe the bias matrix which will be subtracted from any future readings of gyroscope. By using this approach, the constant bias of gyroscope will be decreased or eliminated.

2.2.2 Magnetometer Calibration

The accuracy of a magnetometer is highly dependent on the calculation and subtraction in the software of stray magnetic fields. By convention, these fields are divided into those that are fixed (termed Hard-Iron effects) and those that are induced by the geomagnetic field (termed Soft-Iron effects [3].

A hard-iron offset is resulting from permanently magnetized ferromagnetic components in the smartphone. Since the magnetometer and smartphone rotate together, the hard-iron offset is a simple vector V_{sp}, which adds to the magnetometer reading. Also, the factory calibration will appear as a fixed additive vector V_{sensor}, both vectors are combined as a single hard-iron vector:

$$V = V_{sp} + V_{sensor}$$

Magnetic field Bp measured by a smartphone magnetometer in the absence of hard- and soft-iron effects after rotations in yaw ψ, pitch θ and roll φ by the rotation matrices Rz(ψ), Ry(θ), and Rx(φ) as:

$$Bp = Rx(\varphi)Ry(\theta)Rz(\psi)Br = Rx(\varphi)Ry(\theta)Rz(\psi)B\begin{bmatrix} \cos\delta \\ 0 \\ \sin\delta \end{bmatrix}$$

Br is the local geomagnetic field vector with magnitude B and magnetic inclination δ at the smartphone location.

The Magnetic field including hard-iron vector is shown in the equation below:

$$Bp = Rx(\varphi)Ry(\theta)Rz(\psi)B\begin{bmatrix} \cos\delta \\ 0 \\ \sin\delta \end{bmatrix} + V$$

Soft-iron effect as the interfering magnetic field induced by the geomagnetic field onto normally unmagnetized ferromagnetic components in the smartphone.

The magnetic field measured Bp including soft-iron and hard iron:

$$Bp = WRx(\varphi)Ry(\theta)Rz(\psi)B\begin{bmatrix} \cos\delta \\ 0 \\ \sin\delta \end{bmatrix} + V$$

The solution implemented in the app is the four-parameters calibration presented in reference [4]. The objective is to calculate the hard iron error $V = [Vx, Vy, Vz]$ and the magnetic field strength B.

The user requested to do arbitrary movement for the smartphone to determine the locus of the magnetometer. The same locus of magnetometer measurements represents the primary information available to the calibration algorithms to determine the hard- and soft-iron calibration.

2.3 The 3D Orientation Algorithm

When the app is started, it determines the user's initial orientation relative to Earth by analyzing the magnetic field and gravity data via a Direction Cosine Matrix (DCM) [4]. This initial orientation serves as the origin to which future changes in location are added. The step detector begins counting steps based on the step detection algorithm (see section 2.1). When a step is detected, distance traveled is calculated by applying the stride length the calculated in the stride length estimation algorithm (see section 2.1) to the step taken. The gyroscope continuously monitors angular rotations via another DCM [5], to track changes in heading. Together, the change in heading and distance traveled define a position vector which is applied to the last recorded location to calculate the current location. The current position is plotted to an on-screen graph and stored in various data files for later use.

The calculated biases are subtracted from gyroscope and magnetometer to have a bias-free data. To increase the accuracy and mitigate the dynamic error we implement a two-step Kalman filter algorithm. Details in the next section.

2.4 2-steps Kalman Filter

The idea of two-step Kalman filtering is from the paper [6]. The equation used in the app is derived from both papers [7], and [8].

To Be continue …

Chapter 3 Results

To Be continue …

Chapter 4 Conclusion

To Be continue …

References

[1]   A. Rodriguez and U. Shala, "Indoor Positioning using Sensor-fusion in Android Devices," *Sch. Heal. Soc. Dep. Comput. Sci. Embed. Syst.*, no. September, p. 58, 2011, doi: 10.1.1.367.2683.

[2]   N. Strozzi, F. Parisi, and G. Ferrari, "A Novel Step Detection and Step Length Estimation Algorithm for Hand-held Smartphones," *IPIN 2018 - 9th Int. Conf. Indoor Position. Indoor Navig.*, no. September, pp. 24–27, 2018, doi: 10.1109/IPIN.2018.8533807.

[3]   T. Ozyagcilar, "Calibrating an eCompass in the presence of hard and soft-iron interference," *Free. Semicond. Ltd*, pp. 1–17, 2012, [Online]. Available: http://www.freescale.com/files/sensors/doc/app_note/AN4246.pdf.

[4]   T. Ozyagcilar, "Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors," *Free. Semicond. AN*, pp. 1–21, 2012, [Online]. Available: http://freescale.com.hk/files/sensors/doc/app_note/AN4248.pdf.

[5]   O. J. Woodman, "Number 696 An introduction to inertial navigation An introduction to inertial navigation," 2007, Accessed: Jul. 31, 2020. [Online]. Available: http://www.cl.cam.ac.uk/http://www.cl.cam.ac.uk/techreports/.

[6]   P. Patonis, P. Patia, I. N. Tziavos, D. Rossikopoulos, and K. G. Margaritis, "A fusion method for combining low-cost IMU/magnetometer outputs for use in applications on mobile devices," *Sensors (Switzerland)*, vol. 18, no. 8, 2018, doi: 10.3390/s18082616.

[7]   K. Feng *et al.*, "A new quaternion-based kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm," *Sensors (Switzerland)*, vol. 17, no. 9, 2017, doi: 10.3390/s17092146.

[8]   G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *In Pract.*, vol. 7, no. 1, pp. 1–16, 2006, doi: 10.1.1.117.6808.