A Sensor-based Indoor Positioning Algorithm for Smart-Tour Applications

Vinh Tran-Quang*, Phung Dinh Phuc
Hanoi University of Science and Technology – No. 1, Dai Co Viet Str., Hai Ba Trung, Ha Noi, Viet Nam
Received: May 29, 2018; Accepted: June 29, 2018

Abstract

This article presents an indoor positioning algorithm using accelerometer, gyroscope, and magnetic sensors to determine the location of travel users, to provide location-based services for users on intelligent travel guide systems. The main contributions of the proposal method on this paper are the application of a Kalman filter to eliminate interference of sensor signals that are received from integrated sensors on common smartphones, to handle the time velocity drift when user moving smartphone while standing in place, and therefore, to increase the positioning accuracy. The experiment results show that the proposed positioning algorithm achieves decimeter accuracy in indoor environment, which is suitable for positioning applications such as smart-tour applications.

Keywords: Indoor Positioning Algorithm, Location-based Services, Kalman Filter

1. Introduction

The most common and widespread positioning method today is the Global Positioning System (GPS). Unfortunately, GPS is not an effective way for positioning in indoor environments such as inside buildings or underground works. Sensor-based indoor positing techniques can be used for applications which require high accuracy in indoor environments such as automated travel guides, navigation in the shopping center. In this article, we propose and develop an indoor positioning algorithm that exploited three types of basic sensors integrated in common smartphones: accelerometer, gyroscope, and magnetic sensors to determine the location of travel users. To achieve accuracy requrement in a sensor-based indoor positioning, it is necessary to solve some problems such as eliminating the interference, eliminating cumulative errors. Moreover, the proposed method must be simple, stable and consume less resources of smartphones.

2. Preliminary

2.1 Accelerometer Sensor

Accelerometer is an electromechanical device that measures proper acceleration value as acceleration in three axes. The acceleration is the measurement of the change in velocity or speed divided by time. Using accelerometer sensor can measure the movement of the object according to the corresponding axis (Fig. 1a). For example, the accelerometer detects the direction of the smartphone and rotates the screen according to landscape or portrait mode.

2.2. Gyroscope Sensor

Gyroscope Sensor, also known as angular rate sensor, is a device that senses angular velocity. In simple terms, the angular velocity is the change in rotational angle per unit of time. Angular velocity is generally expressed in deg/s (degree per second). The gyroscope is used in smartphones to find the location and orientation of the device. Similar to the accelerometer, the measured value of gyroscope sensor is taken from the x, y, z-axes as shown in Fig. 1b. Combined the gyroscope with an accelerometer sensor allows the device detecting the motion of six axes: left, right, up, down, forward and back as well as pitch, roll, and yaw for more accurate motion sensation.

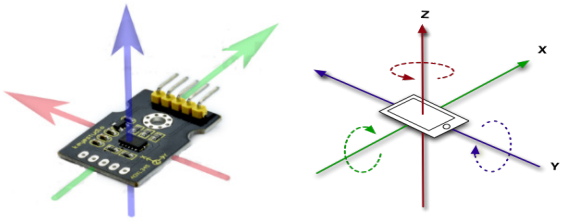

Fig. 1. Accelerometer and gyroscope sensor.

2.3. Magnetic Sensor

A magnetic or compass sensor is a device for detecting and measuring magnetic fields. It provides mobile phones with a simple orientation in relation to the Earth's magnetic field. Under the influence of noise, smartphones do not use the conventional magnetic compass, which uses a different technology to determine the direction of the device. Especially, smart-

* Corresponding author: Tel: (+84) 912636939
Email: vinh.tranquang1@hust.edu.vn

phones will measure signals with extremely low frequency coming from a certain direction (South or North). With the help of the accelerometer sensor, the compass sensor of smartphones can give an accurate orientation of users. The principle of magnetic sensor is based on the Hall effect discovered by Hall in 1879. The effect is based on the interaction between moving electric carriers and an external magnetic field. In metal, these carriers are electrons. When an electron moves through a magnetic field, upon it acts a sideways force [1].

2.4. Kalman Filter

2.4.1. Introduction to Kalman Filter

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean squared error [2]. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown. The Kalman filter estimates a process by using a form feedback control. Accordingly, the filter evaluates the state of the process at a time after the response from the (noise) measurements. As such, the Kalman filter equations are divided into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback, i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrects equations. Indeed, the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems.

2.4.2. The discrete Kalman filter

The Kalman filter addresses the general problem of trying to estimate the state $x \in R^n$ of a discrete time-controlled process that is governed by the linear stochastic difference equation:

$$\mathbf{X}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \mathbf{B}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1},$$

with a measurement $z \in R^m$ that is

$$\mathbf{z}_k = \mathbf{H}\boldsymbol{x}_k + \boldsymbol{v}_k.$$

The random variables w and v represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions p(w) ~ N(0,Q) and p(v) ~ N(0,R). In practice, the process noise covariance Q and measurement noise covariance R matrices might change with each time step or measurement, however here are assuming they are constant. The $n \times n$ matrix A relates to the state at the previous time step $k-1$ to the state at the current step k, in the absence of either a driving function or process noise. The $n \times 1$ matrix B relates to the optional control input ... to the next state x. The $m \times n$ matrix H relates the state to the measurement z_k.

In the actual implementation of the filter, the measurement noise covariance R is usually measured prior to operation of the filter. Measuring the measurement error covariance R is generally practical (possible) because we need to be able to take some off-line sample measurements in order to determine the variance of the measurement noise. The determination of the process noise covariance Q is generally more difficult as we typically do not have the ability to directly observe the estimating process. Sometimes a relatively simple (poor) process model can produce acceptable results if one "injects" enough uncertainly into the process via the selection of Q. Certainly, in this case, one would hope that the process measurements are reliable. In either case, whether or not we have a rational basis for choosing the parameters, often times superior filter performance (statistically speaking) can be obtained by tuning the filter parameters Q and R. The tuning is usually performed off-line, with the help of another Kalman filter in a process generally referred to as system identification [3].

2.4.3. The Kalman filter for sensor signals

The sensors signal from the smartphone is affected by many noises (e.g., the process noise and measurement noise). We will design a Kalman filter to estimate the best value of these values. To create the Kalman filter we need to define parameters in the equations of the filter. With this case, we can see the system does not affect the sensor signals, the control input matrix B and the control vector u are equal to 0. We assume that the value of the current signal is equal to the signal at the previous time, so the state transition matrix A = 1. In fact, the matrix H also changes but in this case is the singularity input so we set H = 1 to simplify the problem. There are the equations of Kalman filter:

Time Update:

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1;}$$

$$\mathbf{P}_k = \mathbf{P}_{k-1} + \mathbf{Q}$$

Measurement Update:

$$\mathbf{K} = \mathbf{P}_k / (\mathbf{P}_k + \mathbf{R})$$

$$\mathbf{X}_k = \boldsymbol{x}_k + \mathbf{K}(\boldsymbol{z}_k - \boldsymbol{x}_k)$$

$$\mathbf{P}_k = (1 - \mathbf{K}_k)\mathbf{P}_k$$

In order to filter the noise for the sensor signals, choosing the process noise covariance Q is not easy. It is necessary to go through many experiments and use the statistical method to obtain the filtered signal as desired. After several experiments, we set the process noise covariance Q = 0.001. The choice of the measurement noise covariance R affects the system's estimated speed. According to the equation for calculating the coefficient K, we can see that K is inversely proportional to R. If the value R is large then the estimation speed is slower, and the estimated value seems less reliable than the measured value. In contrast, the smaller value R, the estimation speed is faster and more reliable. So, choosing the measurement noise covariance R is quite important. It is possible to set R as a fixed value or a mutable value. In this paper, we select the measurement noise R approximately 1% of the measured value (R = 0.01).

3. The Proposal Sensor-based Indoor Positioning Algorithm

3.1 Diagram of the Proposal Algorithm

The implementation of the proposal algorithm is shown in Fig. 2. Input data is the values received from the accelerometer sensor, gyroscope sensor and the initial direction from the magnetic sensor and estimate the direction of gravity using current quaternion. After that, we normalize the accelerometer data according to the acceleration of Earth's axis. Then we use the integral to obtain the velocity. Because velocity is always drifted over time due to the affection of noises so we need to create a method to eliminate the drift. Finally, an integral is used to calculate the corresponding coordinate and update the use position step-by-step.

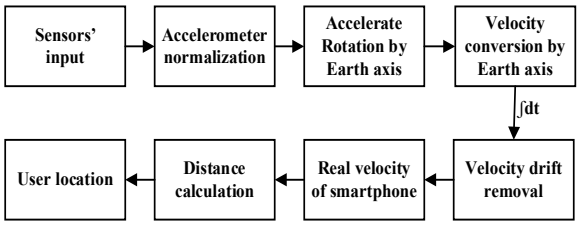

Fig. 2. Diagram of the proposal algorithm.

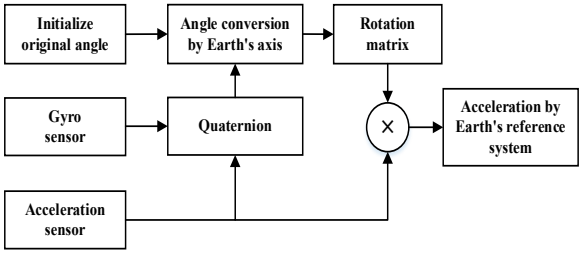

Fig. 3. Acceleration normalization process.

3.2. Acceleration Normalization

The received data from the accelerometer sensor is according to three axises Ox, Oy, Oz of the device. To normalize data in the acceleration of Earth's axis we follow the process as shown in Fig. 3. Step 1: Initializing the original angle using the magnetic sensor. Step 2: Fusing data from the gyroscope sensor and accelerometer to calculate the quaternion matrix. Step 3: Updating the angle and direction with the quaternion. Step 4: Calculating the rotation matrix from the angle and direction. Step 5: Multiplying the rotation matrix with acceleration to obtain the acceleration by Earth's reference system.

3.2.1. Quaternion

The development of Quaternion is attributed to W. R. Hamilton in 1843. Quaternions are very efficient for analyzing situations where rotation in $\boldsymbol{R^3}$ space are involved. A quaternion is defined as below:

$$q = q_0 + q_1 i + q_2 j + q_3 k = q_0 + q^{\omega}$$

$$i^2 = j^2 = k^2 = -1$$

$$ij = -ji = k$$

$$jk = -kj = i$$

$$ki = -ik = j$$

According to the Quaternion's theorem 1 [4] we can see that for any quaternion unit:

$$q = q_0 + q = \cos\frac{\theta}{2} + v\sin\frac{\theta}{2}. \quad (1)$$

And for any vector $\boldsymbol{v} \in \boldsymbol{R^3}$, the action of the operator

$$L_q(v) = qvq^* = (q_0^2 - \|q\|^2)v + 2(q.v)q + 2q_0(q \times v) \quad (2)$$

on $\boldsymbol{v}$ is equivalent to a rotation of the vector through an angle θ about $\boldsymbol{u}$ as the axis of rotation. Let p and q be two unit quaternions. We first apply the rotation operator Lp to the vector u and obtain the vector v. Continuing the rotation operator Lq on the vector v we then obtain the vector w. Equivalently, we apply the composition Lq◦Lp of the two operators [4]:

$$w = L_q(v) = q.v.q^* = q.(p.u.p^*).q^* = (q.p).u.(q.p)^* = L_{qp}(u) \quad (3)$$

3.2.2. Rotation Matrix

A rotation matrix is a matrix that is used to perform a rotation in Euclidean space. The new coordinates *(x',y')* of the point *(x,y)* after rotation are:

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

Write in the matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

In three dimensions, when rotating an angle θ, the rotation matrix form is:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For example, rotating the vector (1, 0, 0) around the axis z of $\theta = 90^0$ can be represented by a matrix:

$$R_z(90^\circ) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} cos90^\circ & -sin90^\circ & 0 \\ sin90^\circ & cos90^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Other rotation matrices can be obtained from these three using matrix multiplication:

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma)$$

represent a rotation which yaw, pitch and roll angles are α, β, γ respectively.

3.3. Velocity Drift Elimination

The acceleration signal is always unstable due to the influence of many factors that lead to the drift of estimated velocity. In addition, the velocity is calculated by the acceleration integral so accumulation error is quite large. The consequence is that within a short time the position of an object is significantly altered even the subject is not moving. So, we need to reduce the drift of velocity in order to improve the accuracy of the algorithm. It must be based on the characteristics of each system. In this case, we use the step detection algorithm to handle the drift.

3.3.1. Step Detection

By detecting the user's steps, we will reset the velocity value $v(t) = 0$. The block diagram of the step detection algorithm is shown in Fig. 4 [5].

Step 1: Use the measured acceleration values, calculate the average value of the acceleration by the formula: a = $\sqrt{(ax)^2 + (ay)^2 + (az)^2}$.

Step 2: Filter the noise of the average acceleration to smooth out the data by the Kalman filter.

Step 3: If the filtered average acceleration is less than a pre-decided threshold value, we will update the velocity to zero.

Step 4: Repeat from step 1 until all velocity data samples are updated.

Step 5: Combine the correct v(t) during the stationary periods and v(t) during non-stationary periods together to yield zero-velocity updated v(t).

Fig. 5 shows the graph of average acceleration when user is moving. The blue line (the Norm line) represents the average acceleration signal. The green line is the threshold of acceleration (a_{th} = 0.5, determined experimentally). When user is standing or feet are placed on the ground, we can see that the average acceleration value is less than the threshold. In some steps, there are many times the average acceleration is lower than the threshold (red circles in Fig. 5) but it is not the moment when user completes the step. To solve this issue, it is necessary to give the data through the designed Kalman filter. Fig. 6 shows the results of step detection algorithm using the Kalman filter. The green graph is the average acceleration signal before filtering. The red graph represents the filtered average acceleration signal.

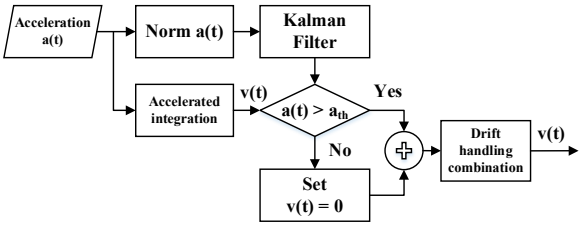

Fig. 4. Diagram of step detection algorithm.

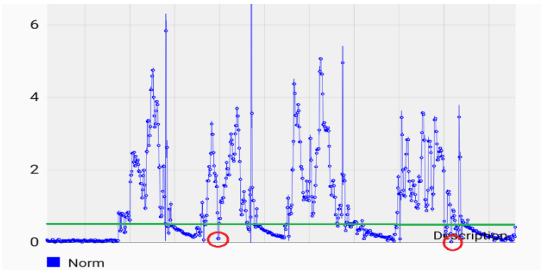

Fig. 5. Raw data of accelerometer sensor.

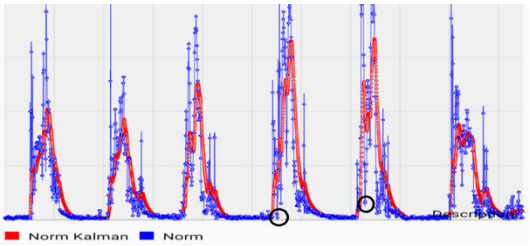

Fig. 6. Acceleration before and after filtering.

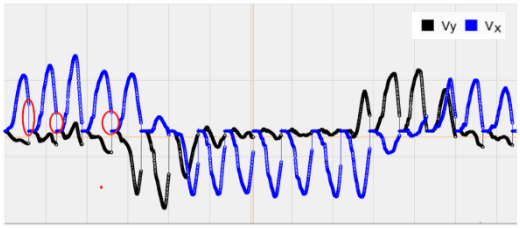

Fig. 7. Velocity with drift.

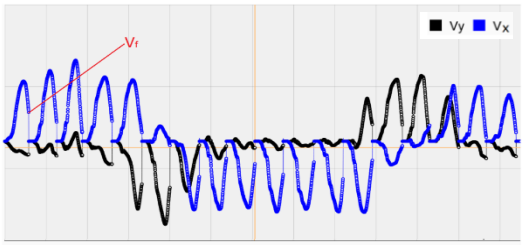

Fig. 8. Velocity after use is removing drift.

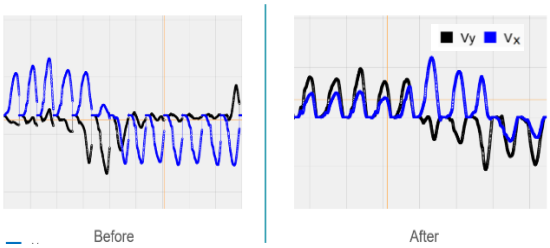

Fig. 9. Results of enhancement drift removal.

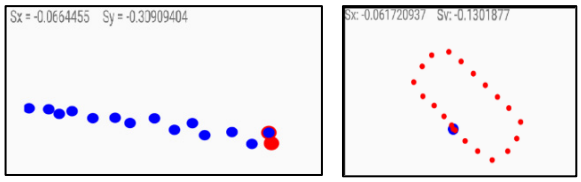

Fig. 10. Trajectory of user on test scenarios.

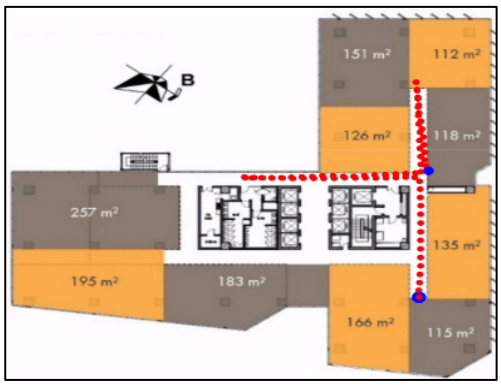

Fig. 11. Positioning in smart-tour application.

Because the only moment when the user completes a step the velocity is set to zero, the drift still exists during that step as we can see in Fig. 7 (red circled). Therefore, the drift caused by integration must be removed.

3.3.2. Enhancement Drift Removal

At each moment, the velocity value is drifted a certain value which accumulates after each footstep. The algorithm for enhancement drift removal is implemented as in the following process:

Step 1: At each user's footstep, detect the moment when the user puts his foot on the ground and determine the velocity at this point as v_f (Fig. 8).

Step 2: Calculate the drift rate by dividing v_f by the total of samples on each footstep.

Step 3: At each sampling cycle of the footstep, calculate the drift velocity by multiplying the drift rate D by the index I of the cycle: $v_{drift} = D \times i$.

Fig. 9 shows the result of the step detection algorithm that combines the enhancement drift removal, whereby the drift velocity is removed.

4. Experimental Results

To evaluate the accuracy of the proposed algorithm, we conduct variety of test scenarios. A user holding a smartphone that have been installed the proposed sensor-based indoor positioning algorithm.

4.1. Scenario 1: User Goes Straight

In this scenario, user goes straight forward and back to the original position (Fig. 10a). The start coordinates of the user are (x,y)=(0.0,0.0), the user's coordinates after moving back are (x,y)=(0.06,0.3) meter. The error for x axis is 6 cm, for y axis is 30 cm.

4.2. Scenario 2: User Moves Around

The next scenario, user moving around in larger area. As in scenario 1, the user start coordinates are (x,y) = (0.0,0.0). After moving a loop, the end coordinates are (x,y) = (0.06, -0.13) meter. The error for x axis is 6 cm, for y axis is 13 cm (Fig. 10b).

4.3. Scenario 3: User Moves Freely

To verify the accuracy and applicability of the proposed positioning algorithm for our smart-tour application, the results are plotted in Fig. 11. From the start point, the user walks through corridors and the mobile application will mark the user's location corresponding to the actual one. A blue dot indicates the current location of the user, the red dots are the position of steps measured.

5. Conclusion

Experimental results show that the proposed algorithm has positioned the user with high precision and small error. Also, the problems encountered such as the drift velocity has been resolved thoroughly.

Acknowledgments

This article is supported by the project 15/2017/HD-KHCN-DT, “Research and development of interactive software system, database and website for intelligent tourism guide in Tuyen Quang province.”

References

[1] Bort-Roig, J., Gilson, N.D., Puig-Ribera, A. et al., “Measuring and influencing physical activity with smartphone technology: a systematic review,” Sports Med, Vol. 44(5), pp 671–686, 2014.

[2] Kalman RE and Bucy RS, “New results in linear filtering and prediction theory,” ASME. J. Basic Eng., vol. 83(1), pp. 95-108, 1961.

[3] Welch, Greg & Bishop, Gary, "An introduction to the Kalman filter," Proc. Siggraph Course, vol. 8, 2016.

[4] Quaternions and Rotations (Com S 477/577 Notes) Yan-Bin Jia Sep 5, 2017.

[5] Ruoyu Zhi, “A drift eliminated attitude & position estimation algorithm in 3D” (2016), Graduate College Dissertations and Theses, paper 450.