# Short-Term Motion Tracking Using Inexpensive Sensors

Filip Matzner(✉) and Roman Barták

Faculty of Mathematics and Physics, Charles University in Prague,
Malostranské náměstí 25, 118 00 Praha 1, Czech Republic
floop@floop.cz, bartak@ktiml.mff.cuni.cz

**Abstract.** Current consumer electronics is equipped with various sensors, among which accelerometer, gyroscope, and magnetometer represent typical examples. In this paper, we study the possibility of using these low-cost sensors for 3D motion and orientation tracking. In particular, we thoroughly describe a simple dead-reckoning algorithm for sensor data fusion which produces a 3D path of the device in real time. More importantly, we propose a method of automated stabilization every time the device stands still, which corrects the bias caused by sensor inaccuracies. This method extends the time when motion tracking is reliable. We evaluate the proposed pipeline in a variety of experiments using two common smartphones.

**Keywords:** Motion tracking · Sensor fusion · Signal processing · Inertial navigation · Visualization · Dead-reckoning · Smartphones

## 1 Introduction

Precise motion tracking is very important in robotics for navigation, however, it can also be exploited in many other areas such as mapping of surrounding environment or taking panoramatic snapshots using smartphones. The question that we ask in this paper is whether low-cost sensors included, for instance, in common smartphones and "toy robots" such as AR.Drone could be used for this purpose. The goal is to design a method for short-term tracking of device movement and orientation using only its on-board sensors – accelerometer, gyroscope, and magnetometer.

The approaches to motion tracking can be divided into two basic categories. The first one is to ignore the behaviour of the device (i.e., the way it moves) and to trust the data received from the sensors. Those are called *non-model methods*. The second one is to create a model of the device behaviour and to trust more the data in accordance with the model than the data contrary to the model. Those are called *model methods*. Model methods are particullary useful for objects with limited means of movement, such as cars and airplanes. For instance, when an airplane is flying forwards, it is highly unprobable that it stops in place and starts moving backwards even if the sensors claim so.


O. Pichardo Lagunas et al. (Eds.): MICAI 2015, Part II, LNAI 9414, pp. 583–601, 2015.
DOI: 10.1007/978-3-319-27101-9_45

Probably the most profound example of the model method is the Kalman filter [7] and its variants. It is a probabilistic method, which plays a key role in the inertial navigation of airplanes, vehicles, and in signal processing in general. It operates on sensor data containing noise and inaccuracies and produces a statistically optimal estimate of variables such as velocity, position, and orientation. With suitable parameters, even multiple contradicting sensors or measurements can improve the overall filter accuracy, therefore it is best used with additional sensors such as odometry [3], strategically placed RFID tags [16], rolling-shutter cameras [11], GPS [5], and ultrasound rangefinders [22]. The Kalman filter itself is well covered in [19,20], however, understanding of this paper does not require its deeper knowledge. Other examples of probabilistic model methods include unscented and extended Kalman filters, particle filters [12], and multi-model techniques [21]. Nonetheless, there also exist non-probabilistic model methods, such as human indoor navigation based on walking style utilizing fuzzy logic [10].

As we want to track the motion of smartphones, which can be moved in all directions and arbitrarily rotated, no reasonably strong model seems to be available. Therefore we decided to implement a straightforward non-model dead-reckoning method based on simple physics and linear algebra. To eliminate some kinds of inaccuracies, such as sensor drift, we propose a novel stabilization routine based on the fact that gravity and magnetic field keep pointing the same direction no matter the orientation and position of the device.

Our work might resemble the work by Neto *et al.* [14], which uses accelerometer, gyroscope, and magnetometer to track the position of a human hand during a common pick-and-place task. To eliminate the accelerometer inaccuracies, Neto *et al.* use the accelerometer data only when there is a significant motion and suspend the process when the hand stands still. We have improved the method by using the standstill moments to not only suspend the process but also to correct the supposed orientation of the device. Similar methods exist [4,6,16], however, they seem to exploit the fact that the object of interest moves on a horizontal plane whereas our approach is applicable to any 3D motion.

In Sect. 2, we briefly present the sensor technology and its problems. In Sect. 3, we thoroughly describe the sensor fusion procedure, first as a theoretical model and second as an outline of a practical implementation. In Sect. 4, we explain the proposed stabilization method in a step-by-step manner. In Sect. 5, all the presented techniques are evaluated in a variety of experiments using two common smartphones. Finally, in Sect. 6, our results are summarized.

## 2 Background on Sensors

The most widely used technology of smartphone sensors is called micro-electromechanical systems (usually denoted as MEMS). The size of these sensors is in the order of micrometers ($10^{-6}\,m$) and the previously mentioned triple of an accelerometer, a gyroscope, and a magnetometer might be manufactured on a single chip called inertial measurement unit (IMU). The low-cost alternatives of such chips mounted in smartphones are able to produce up to hundreds of readings per second.

The minimal requirements for smartphone sensors are low because they are almost exclusively used for detecting the orientation of the device and not its position. The orientation is afterwards used in games, mobile applications supporting both landscape and portrait view etc.

### 2.1 Noise and Bias

The sensors are affected by various sources of noise and bias. A thorough discussion of the sources of accelerometer noise and evaluation of the corresponding filters can be found in work by Khosla *et al.* [8]. The authors collected data from a flying rocket and during off-line data processing, they applied various filters to estimate the impact of individual noise natures. Conclusions are, that most significant sources of noise include bias (i.e., permanent misalignment of the sensor) and improper mounting. Unfortunately, the non-linearity and influence of the temperature on the accelerometer accuracy is not discussed.

It should be noted that there are contradicting opinions on whether smoothing of the accelerometer signal by, for example, Butterworth filter improves the tracking accuracy or not. In works by Baranski *et al.* [2] and Gusenbauer *et al.* [5], both focused on pedestrian navigation, the filter shows promising results. On the other hand, Khosla *et al.* [8], focused on localization of a rocket, demonstrated all but a neglegible impact of the filter on the overall accuracy.

Similar discussion of the various natures of gyroscope noise can be found in [1,3,17]. All the referred works agree on the fact that the gyroscope drift and accuracy is a strong function of temperature. Shiau *et al.* [17] use artificial neural networks to learn and compensate the temperature effect. Chung *et al.* [3], use analytical methods to estimate the non-linearity and temperature effects by higher order polynomials. Both achieve significant improvement over non-compensated data.

The magnetometer appears to be the least reliable sensor of all the three. When the sensor is moving close to, for instance, a Wi-Fi receiver, a mobile phone, or a large metal object, the magnetic field is significantly unstable. This interference can be at least partially eliminated by proper shielding or filtering [1,4].

We are focused on short-term tracking only, therefore we have not dealt with most of the mentioned problems. It was, however, neccessary to calibrate the magnetometer right before experimentation using the procedure provided by its manufacturer[1].

## 3 Sensor Fusion

In this section, we describe the process of fusing the sensor data to get the position of the device. The sequence of positions with their corresponding timestamps will specify the device movement path.

[1] In our case, the Android OS requires launching an application that uses the magnetometer and rotating the device in the "figure 8 pattern".

Before we begin, we have to define two coordinate systems we will be working with. The world frame of reference (world FoR) represents a fixed coordinate system the device was in at the very beginning of the process. The device frame of reference (device FoR), on the other side, is fixed to the device and rotates with it during its motion. Comparison of the two systems is depicted in Fig. 1. The data produced by the sensors are always relative to the device FoR, because the sensors are mounted to the smartphone case. Additionally, we will assume that at the beginning the device was standing still, i.e., the only force measured by the accelerometer in time 0 was gravity.

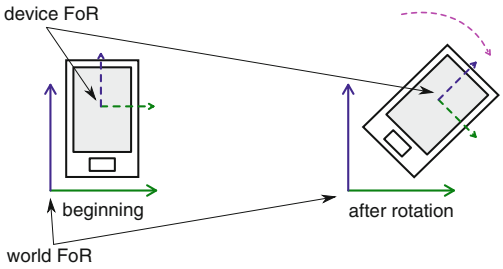



**Fig. 1.** World and device frames of reference.

The data produced by the sensors will be defined as follows:

- **Accelerometer data**, $\text{acc}(t) : \mathbb{R} \to \mathbb{R}^3$ specifies the acceleration vector in $\text{ms}^{-2}$ in time $t$ including the gravity force. Additionally, $\text{acc}(0)$ denotes gravity only.
- **Gyroscope data**, $\text{gyro}(t) : \mathbb{R} \to \mathbb{R}^3$ specifies the rotation rates (i.e., angular speeds) in rad $\text{s}^{-1}$ around the three coordinate axes $x$, $y$, and $z$ in time $t$.
- **Magnetometer data**, $\text{mag}(t) : \mathbb{R} \to \mathbb{R}^3$ specifies the magnetic field strength in $\mu$T along the three coordinate axes in time $t$.

Throughout the paper, we use integration on vector functions. The result of such operation is again a vector function where the integration is performed piecewise as shown in Formula 1.

$$\begin{aligned} f &: \mathbb{R} \to \mathbb{R}^3 \\ f(x) &= \langle f_1(x), f_2(x), f_3(x) \rangle \\ \int f(x)\mathrm{d}x &= \langle \int f_1(x)\mathrm{d}x, \int f_2(x)\mathrm{d}x, \int f_3(x)\mathrm{d}x \rangle \end{aligned} \tag{1}$$

### 3.1 Theoretical Model

We will begin with the simplest possible model, where the device is not rotating during its motion and the sensors are perfectly accurate and provide continuous

data. Because the orientation of the device does not change, the device FoR will have the same orientation as the world FoR during the entire motion. Therefore, we can process the sensor data as if they were relative to the world FoR.

Let us remind some basic physics. Velocity is defined as derivative of position with respect to time (Formula 2) and acceleration is defined as derivative of velocity with respect to time (Formula 3).

$$\begin{aligned} \text{linacc} : \mathbb{R} \to \mathbb{R}^3 &\text{ acceleration in time } t \text{ without gravity} \\ \text{vel} : \mathbb{R} \to \mathbb{R}^3 &\text{ velocity in time } t \\ \text{pos} : \mathbb{R} \to \mathbb{R}^3 &\text{ position in time } t \end{aligned}$$

$$\text{vel}(t) = \frac{\partial \text{pos}(t)}{\partial t} \tag{2}$$

$$\text{linacc}(t) = \frac{\partial \text{vel}(t)}{\partial t} \tag{3}$$

However, in our case, the only known variable is the acceleration, thus we will use the definition the other way around. That is, the velocity is the integral of acceleration (Formula 4) and the position is the integral of velocity (Formula 5).

$$\text{vel}(t) = \int_0^t \text{linacc}(u)\mathrm{d}u \tag{4}$$

$$\text{pos}(t) = \int_0^t \text{vel}(u)\mathrm{d}u \tag{5}$$

Note that the accelerometer sensor also measures gravity which has to be removed before the sensor data can be used in the formulas above. We have assumed that the device stands still at the beginning, thus the gravity can be denoted as acc(0). By putting velocity Formula 4 and position Formula 5 together and subtracting the gravity from the accelerometer data (Formula 6) we can express the position of the device using only the data from the accelerometer (Formula 7).

$$\text{linacc}(t) = \text{acc}(t) - \text{acc}(0) \tag{6}$$

$$\text{pos}(t) = \int_0^t \int_0^t (\text{acc}(u) - \text{acc}(0))\,\mathrm{d}^2u \tag{7}$$

Unfortunately, once we take the rotation of the device into account, the device FoR will rotate with the device, thus accelerometer data from different times will be relative to different coordinate systems. Therefore, the data have to be converted to the world FoR and only after the conversion we can integrate the transformed data the same way we have done in Formula 7. To be able to calculate this conversion, we need to know the orientation of the device.

**Orientation.** Before we define our representation of the orientation, we have to define a 3D rotation matrix [13]. The 3D rotation matrix is used to represent and

perform arbitrary rotation in 3D space. If we have a rotation matrix $A \in \mathbb{R}^{3\times3}$ and we want to apply the rotation it represents to a vector $u \in \mathbb{R}^3$, we can simply pre-multiply the vector by the matrix, i.e., $Au$. The construction of such matrix is, however, a little bit more difficult.

To make the construction easier, we will introduce a known Formula 8 for the rotation matrix representing rotation around an arbitrary axis through an arbitrary angle [13]. For our purposes a slightly simplified version where the axis is a unit vector is sufficient. This formula hardly has an intuitive explanation. However, the procedure that derives it is not that difficult and is clearly explained in [13].

$$R : \mathbb{R}^3 \times \mathbb{R} \rightarrow \mathbb{R}^{3\times3}$$

$$v \in \mathbb{R}^3 \text{ where } v =< a, b, c > \text{ and } \sqrt{a^2 + b^2 + c^2} = 1$$

$$R(v, \theta) \text{ is a rotation matrix through the angle } \theta \text{ around the unit vector } v$$

$$R(\langle a, b, c\rangle, \theta) = \begin{pmatrix} a^2 + (1 - a^2)\cos(\theta) & ab(1 - \cos(\theta)) - c\sin(\theta) & ac(1 - \cos(\theta)) + b\sin(\theta) \\ ab(1 - \cos(\theta)) + c\sin(\theta) & b^2 + (1 - b^2)\cos(\theta) & bc(1 - \cos(\theta)) - a\sin(\theta) \\ ac(1 - \cos(\theta)) - b\sin(\theta) & bc(1 - \cos(\theta)) + a\sin(\theta) & c^2 + (1 - c^2)\cos(\theta) \end{pmatrix} \tag{8}$$

Now, we can define the orientation of the device as the rotation matrix representing the rotation of the device FoR relative to the world FoR (i.e., relative to the initial orientation). The rest of this section describes how to construct such a matrix using the gyroscope data.

It might be tempting to integrate the raw gyroscope data (i.e., angular speeds) the same way we have integrated the accelerometer as shown in Formula 9.

$$\int_0^t \text{gyro}(u)\text{d}u = \ ? \tag{9}$$

The result of such integration are three rotation angles around the axes $x$, $y$ and $z$ representing the rotation of the device from the beginning of the process. Unfortunately, those three angles just by themselves do not provide sufficient information to reproduce the orientation of the device. Even the fact, that rotation composition is not commutative in the 3D space (as depicted in Fig. 2), might give us multiple orientations corresponding to the same three rotation angles. Therefore the result of such integration is effectively useless.

We will solve this problem by sampling the gyroscope data into time intervals, whose duration is getting to zero. We may presume that during each such a short time interval the rotation speed was constant. This presumption allows us to construct rotation matrices representing the rotation of the device during each of these intervals. Finally, we can simply multiply all those matrices to create a matrix representing the rotation from the beginning to the time $t$ as depicted in Fig. 3. This composed matrix will be the orientation matrix.

Let us describe the construction of the orientation matrix in detail. At first, we need the rotation angles representing rotation during a time interval $(t_1, t_2)$. We can calculate these angles by integrating the gyroscope data and we will

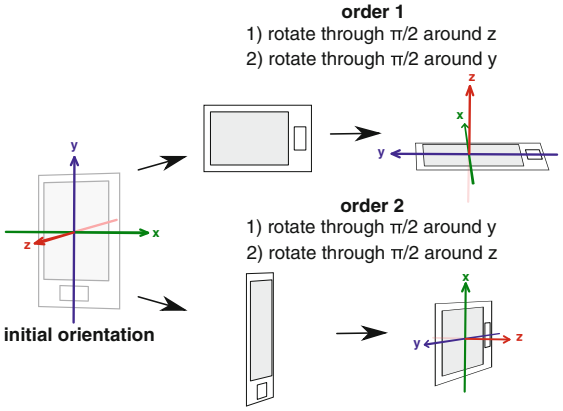



**Fig. 2.** Counterexample for rotation commutativity in 3D

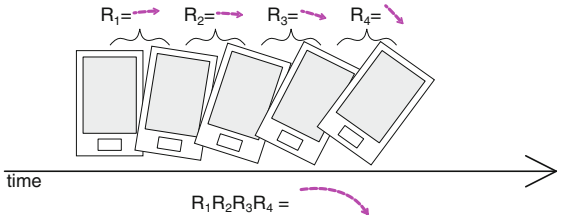



**Fig. 3.** Rotation sampling

denote this vector of rotation angles by the function $\Delta\text{rot}_{\text{ang}}(t_1, t_2)$ defined in Formula 10.

$$\Delta\text{rot}_{\text{ang}} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^3$$
$$\Delta\text{rot}_{\text{ang}}(t_1, t_2) := \int_{t_1}^{t_2} \text{gyro}(t)\text{d}t \tag{10}$$

To convert this vector to a rotation matrix, we will use Formula 8. However, what angle and what axis should we use? Presuming a short time interval, the rotation speed might be considered to be constant. Therefore, the rotation around the coordinate axes $x$, $y$ and $z$ by the angles $\alpha$, $\beta$ and $\gamma$ is the very same as the rotation around the vector $v =< \alpha, \beta, \gamma >$ through the angle $||v||$. Additionally, Formula 8 assumes a unit vector, thus we have to normalize the vector $v$ to a unit vector. It can be achieved by dividing it by its own size, i.e., $v_{norm} = \frac{v}{||v||}$. Using these observations and substituting the $\Delta\text{rot}_{\text{ang}}(t_1, t_2)$ to the rotation matrix Formula 8, we can convert the function $\Delta\text{rot}_{\text{ang}}(t_1, t_2)$ returning a vector to a function $\Delta\text{rot}_{\text{mat}}(t_1, t_2)$ returning a matrix. This conversion is defined in Formula 11.

$$\begin{aligned} \Delta\mathrm{rot}_{\mathrm{mat}} &: \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{3\times 3} \\ \Delta\mathrm{rot}_{\mathrm{mat}}(t_1, t_2) &:= R\left(\frac{\Delta\mathrm{rot}_{\mathrm{ang}}(t_1, t_2)}{||\Delta\mathrm{rot}_{\mathrm{ang}}(t_1, t_2)||}, ||\Delta\mathrm{rot}_{\mathrm{ang}}(t_1, t_2)||\right) \end{aligned} \tag{11}$$

Now we have got everything required to define the function that returns the orientation matrix in a given time $t$. We need to create a rotation matrix out of the gyroscope data as frequently as possible and multiply these rotation matrices up to the time $t$. Formula 12 mathematically expresses this idea and defines the orientation function.

$$\begin{aligned} \mathrm{ori} &: \mathbb{R} \to \mathbb{R}^{3\times 3} \\ \mathrm{ori}(t) &:= \lim_{h\to 0} \prod_{i=1}^{\lfloor t/h \rfloor} \Delta\mathrm{rot}_{\mathrm{mat}}((i-1)h, ih) \end{aligned} \tag{12}$$

where $h$ is the duration between two subsequent samples.

**The Final Formula.** Now we can modify the position function from Formula 7 so it takes the orientation into account. We have the accelerometer data relative to the device FoR and we have the orientation matrix of the device FoR. The only thing we have to do is to pre-multiply the accelerometer data by the orientation matrix to convert them to the world FoR. Formula 13 defines this modified version of the position function.

$$\begin{aligned} \mathrm{pos} &: \mathbb{R} \to \mathbb{R}^3 \\ \mathrm{pos}(t) &:= \int_0^t \int_0^t (\mathrm{ori}(u)\mathrm{acc}(u) - \mathrm{acc}(0))\, \mathrm{d}^2 u \end{aligned} \tag{13}$$

### 3.2 Practical Approach

The theoretical model can be implemented in a straightforward way or utilized in other algorithms for sensor fusion, such as the previously mentioned Kalman Filter.

The straightforward approach is based on Formula 13 and can be implemented as follows. We will process the gyroscope data and the accelerometer data separately. For each gyroscope reading, we integrate the data, convert them to a rotation matrix, and update the current orientation matrix. For each accelerometer reading, we multiply the data by the current orientation matrix (i.e., we convert the data to the world FoR), remove the gravity, and double integrate the result to calculate the position of the device. This implementation is fast enough to run as an online process without the need for high processing power.

The discrete integration itself might be implemented, for instance, as summation of values or the values can be approximated by a function whose integral can be easily calculated. We, however, recommend using the trapezoidal integration

rule [9], as it improves the results significantly [8] and is very easy to implement. The trapezoidal integration is similar to the standard Euler's integration method which approximates a function by rectangles, however, the remaining space between the rectangle and the function is further approximated by a triangle, as depicted in Fig. 4.

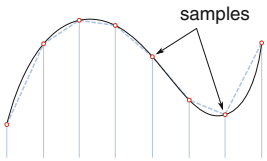



**Fig. 4.** Trapezoidal integration rule

## 4 Automated Stabilization

Even though gyroscope data are usually very accurate, the integration introduces drift, partially because of noise and partially because of insufficient sampling rate. Why is it such a problem? Because if the device orientation is just a few degrees off, Earth's gravity is removed from a wrong direction. If it creates a false acceleration of, for example, $0.1\,\mathrm{ms}^{-2}$ then after ten seconds it pulls the device 10 m off (calculation in Formula 14).

$$\int_0^{10}\int_0^{10} 0.1\mathrm{d}^2 t = 10 \tag{14}$$

Fortunately, the Earth's gravity and magnetic field point a constant direction no matter the orientation and position of the device. We will use this property whenever the device is at rest[2], to adjust the orientation by comparing gravity and magnetic field vectors measured at the beginning with the current values of accelerometer and magnetometer.

### 4.1 Stillness Detection

The device is at rest, when it does not rotate and the only force measured by the accelerometer is gravity. To detect such a moment, we will utilize the high sensitivity of the gyroscope and implement something called an *angular rate energy detector* [18]. The implementation is as follows. Whenever squared magnitudes of the gyroscope data stay below a specified threshold $\alpha$ during a specified time window of length $W$, we will consider the device to be still (Formula 15). The selection of the threshold and the size of the time window

[2] The technique of updating the stored values when the device is at rest is usually called zero-velocity update [18].

depends on the quality of the sensor and various other aspects, such as whether the device lays on a table or is held in a shaking hand.

$$\text{still}(t) \Leftrightarrow \frac{1}{W} \int_{t}^{t+W} \|\text{gyro}(u)\|^2 du < \alpha \tag{15}$$

In theory, this technique would detect stillness, for instance, while moving in a straight line at a constant speed. In practice, however, it is very difficult to move the device in a way the gyroscope data would seem still. For a thorough discussion on various stillness detectors please refer to [18].

### 4.2 Zero Velocity

Even when we stop the motion and the device is at rest, the drift could cause that acceleration is not integrated to zero. It can be a serious problem because the integrated acceleration represents velocity and thus the algorithm behaves as if the device was moving even though it rests in place as depicted in Fig. 5. Therefore, it is very important to set the supposed velocity to zero whenever the device stands still.

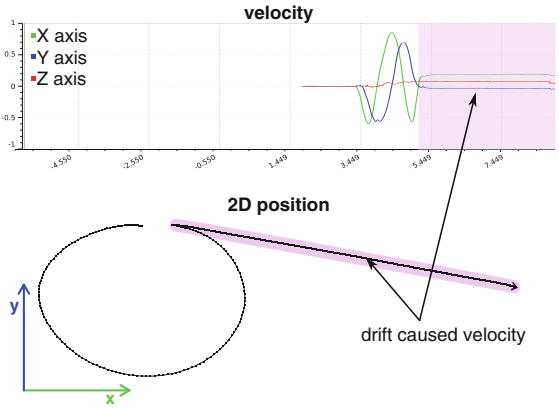



**Fig. 5.** Device is moved in a shape of a circle and then put to rest. The integrated false velocity keeps dragging the supposed position off.

### 4.3 Gravity Fix

When the device is still, the current acceleration vector in the world FoR should match the gravity vector measured at the beginning. If not, we will adjust the stored orientation to make it true as seen in Fig. 6.

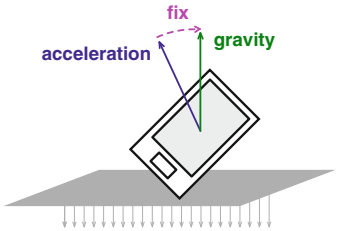



**Fig. 6.** Matching the acceleration to the gravity vector

Let's assume that the device is standing still and we have the following data:

- $gra \in \mathbb{R}^3$ is the gravity measured at the beginning.
- $acc \in \mathbb{R}^3$ is the vector currently measured by the accelerometer, including gravity. Since the device is still, it is nothing but the gravity in the device FoR.
- $ori \in \mathbb{R}^{3\times3}$ is the current orientation matrix.

Then we can fix the orientation drift in two steps:

1. Create a rotation matrix $fix \in \mathbb{R}^{3\times3}$ that rotates the vector $ori \cdot acc$ (i.e., acceleration in the world FoR) to match the vector $gra$ (i.e., acceleration at the beginning). Formula 16 expresses this idea mathematically.

$$fix \cdot ori \cdot acc = gra \tag{16}$$

But how does a matrix that rotates a vector $a \in \mathbb{R}^3$ to match a vector $b \in \mathbb{R}^3$ look like? We will use the definition of the rotation matrix around an arbitrary unit vector through an arbitrary angle (Formula 8), but we need to find the vector to rotate around and the angle to rotate through. A very simple way of finding both is to use a known Formula 17 for cross product of two vectors [9]. Visual representation of the formula is depicted in Fig. 7.

$$\begin{aligned} &a, b \in \mathbb{R}^3 \\ &\quad \theta \text{ is the angle between } a \text{ and } b \\ &\quad n \text{ is a unit vector perpendicular to the plane} \\ &\qquad \text{generated by } a \text{ and } b \\ &a \times b = \|a\|\|b\| n \sin\theta \end{aligned} \tag{17}$$

Having the definition of the cross product, it is not difficult to calculate the vector and the angle to be substituted into the definition of the rotation matrix (8). The result is a matrix that rotates a vector $a$ to match a vector $b$, as shown in Formula 18.

$$R_{match}(a, b) := R\left(\frac{a \times b}{\|a \times b\|}, \arcsin\left(\frac{\|a \times b\|}{\|a\|\|b\|}\right)\right) \tag{18}$$

Having Formula 18, we can simply substitute appropriate vectors $ori \cdot acc$ and $gra$ for the $a$ and $b$ to define the matrix $fix$ mentioned in Formula 16. The result is shown in Formula 19.

$$fix := R_{match}(ori \cdot acc, gra) \tag{19}$$

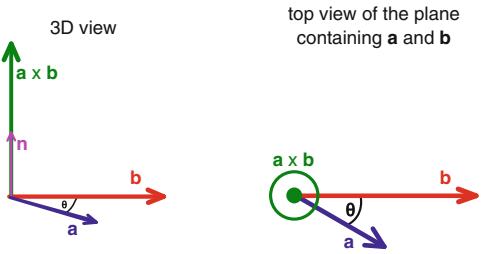



**Fig. 7.** Cross product visual representation

2. Pre-multiply the orientation matrix by the $fix$.

It should be mentioned that the magnitude of the Earth's gravity ($\approx 9.81 \text{ms}^{-2}$) should never be considered to be a constant, since the accelerometer sensitivity varies with the sensor orientation relative to the measured force. For instance, the devices evaluated in our experiment exhibited 1 - 1.5 $\text{ms}^{-2}$ difference in the Earth's gravity measurement when the device is simply turned over.

### 4.4 Magnetic Field Fix

Even when the orientation of the device is fixed using the gravity vector, it still remains ambiguous as demonstrated in Fig. 8. To eliminate this ambiguity, we will use the magnetometer sensor. It might be tempting to repeat the same steps as we have done with the gravity fix, but this time with the magnetic field. Such an approach, however, could break the previous gravity fix and alter the orientation matrix in a way that the accelerometer data in the world FoR would not perfectly match the gravity measured at the beginning. In such a case, the gravity would again be removed from a wrong direction and that, as we have previously calculated, is a serious flaw. Therefore, after this process of fixing the orientation by magnetic field, we still require the acceleration vector in the world FoR to point the very same direction as the gravity.

The only way of adjusting the orientation matrix with the persistency of the accelerometer vector in the world FoR, is to perform the rotation around the accelerometer vector itself. To have the fix as accurate as possible at the same time, we will choose the rotation angle minimizing the distance between the current magnetic field vector in the world FoR and the magnetic field vector

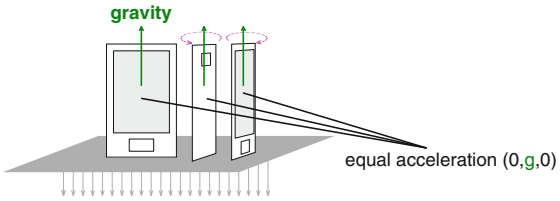



**Fig. 8.** The device measures the same acceleration in multiple orientations

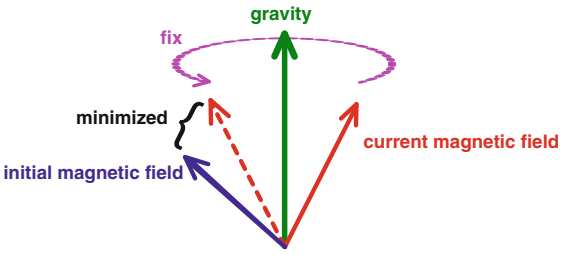



**Fig. 9.** Rotating around the gravity vector so that the initial and current magnetic vectors are as close as possible

measured at the beginning. The idea is depicted in Fig. 9, where the accelerometer vector in the world FoR is denoted as gravity as they are equal, thanks to the previously described gravity fix.

Such a fixing matrix can by constructed using just the tools we already have. Let's assume that the device is standing still, we have already fixed the orientation using gravity, and we have the following data:

- $mag_b \in \mathbb{R}^3$ is the magnetic field measured at the beginning.
- $mag_c \in \mathbb{R}^3$ is the vector currently measured by the magnetometer.
- $gra \in \mathbb{R}^3$ is the gravity measured at the beginning and the accelerometer vector in the world FoR at the same time.
- $ori \in \mathbb{R}^{3\times3}$ is the current orientation matrix.

The steps to fix the orientation are as follows:

1. Project the $mag_b$ and $mag_c$ vectors in the world FoR to the plane perpendicular to the gravity vector[3]. Denote the results as $pmag_b$ and $pmag_c$. It is equal to viewing the situation from such position that the gravity vector points right towards us, as demonstrated in Fig. 10.
2. Create a rotation matrix $fix$ that rotates the vector $pmag_c$ to the vector $pmag_b$. We can observe that the rotation matrix has to represent a rotation around the gravity vector.
3. Pre-multiply the orientation by the $fix$.

[3] We can achieve the projetion by creating a rotation matrix that rotates the gravity vector to the $z$ axis, then rotate the $mag_b$ and $ori \cdot mag_c$ vectors using this matrix, set their $z$ coordinate to zero and rotate them back.

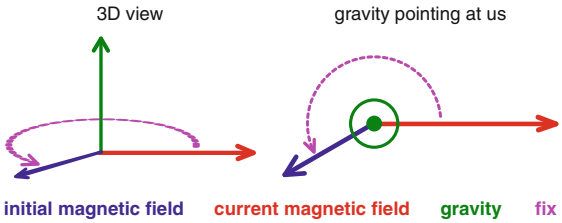



**Fig. 10.** The vectors from Fig. 9 after projection to the plane perpendicular to the gravity

## 5 Experimental Evaluation

We have implemented a computer software, which provides a user-friendly interface for processing and visualization of data produced by accelerometer, gyroscope, and magnetometer sensors (a screenshot is depicted in Fig. 11). The fusion method is based on the straightforward algorithm from the section Practical Approach and enhanced by the automated stabilization. We will perform a series of experiments with this software to demonstrate the properties of the algorithm.

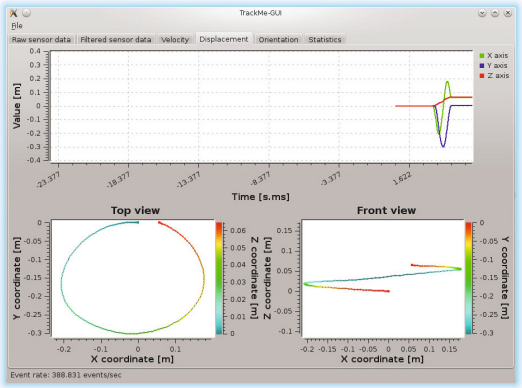



**Fig. 11.** TrackMe GUI screenshot

The situation will always be visualized from two angles, as demonstrated in Fig. 12. The top view observes the device from the direction of the $z$ axis and the front view observes the device from the direction of the $y$ axis. The color of each tracepoint will denote the distance from the viewer, with red being the closest and purple being the furthest. The entire spectrum is depicted in Fig. 13.

The first experiment will be performed on two different smartphones - Huawei Honor U8660 and Samsung Galaxy S III. We have chosen these particular devices

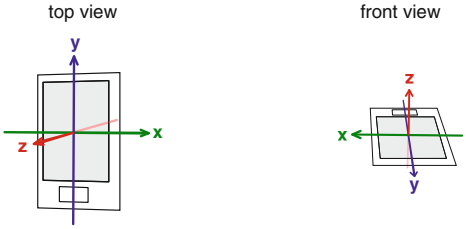



**Fig. 12.** Two views of a smartphone on a table


**Fig. 13.** Distance spectrum

for their significant difference in price and performance. Table 1 gives a comparison of sensor reading rates for both smartphones.

During the first experiment, the device will lay on a table and we will move it in the shape of a circle. The result is depicted in Fig. 14. The movement took less than three seconds and the result is surprisingly accurate. Especially for the Galaxy phone, the shape is smooth from both views. The Honor phone suffers from lower sensor reading rate and the motion path slightly drifts. From now on, we will perform the experiments with the Galaxy phone only.

Let us prolong the motion duration and draw the same circle three times in a row. The result drawn in Fig. 15a demonstrates that with longer motion the shape can no more be recognized because of the gyroscope drift and consequent false acceleration. The first circle is clearly visible but even the next one is off by a diameter. The reason is, that we have not given the stabilization system a chance to fix the drift and the result would actually be the same even if the stabilization would not be implemented at all.

To see the benefits of the automatic stabilization, we will insert a small pause (cca. one second) after each circle. During this pause, the software detects stillness and updates the orientation matrix. Figure 15b shows the same three circles with two pauses in between. The motion took twelve seconds and the Galaxy phone, again, draws a smooth and accurate picture. It has all the circles clearly visible and the starting position is almost the same as the ending position.

**Table 1.** Number of readings per second

| | Samsung Galaxy S III | Huawei Honor U8660 |
|---|---|---|
| Accelerometer | 105 | 52 |
| Gyroscope | 206 | 61 |
| Magnetometer | 102 | 95 |
| **sum** | **413** | **208** |

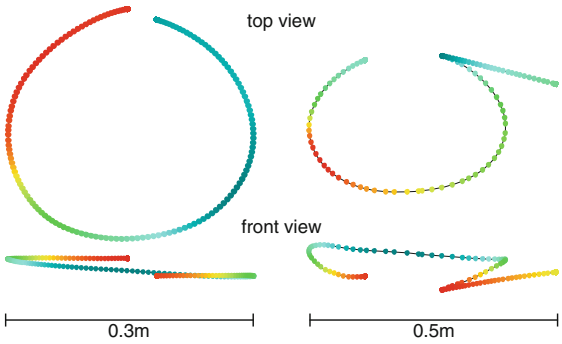
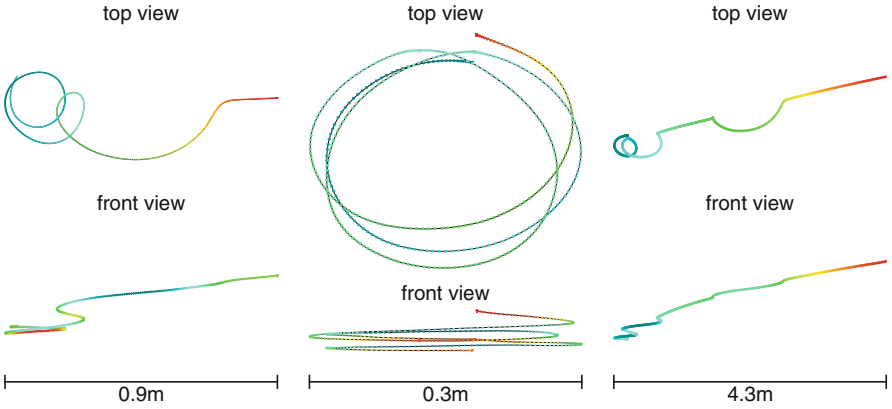



**Fig. 14.** Circle shape - 30 cm in diameter - Galaxy S III (left) and Huawei Honor (right)

To compare this result to the case where the stabilization system is turned off, we have performed the same experiment once again, but we have disabled all the corrections introduced in the section Automatic Stabilization. The drawing is demonstrated in Fig. 15c. Not surprisingly, the results are even worse than in the experiment without pauses (as shown in Fig. 15a), because the double integration of false acceleration introduces a drift whose strength rises quadratically with time. The last circle was drawn after twelve seconds of motion and ended up as nothing but a little curve in a fast nonexistent movement.



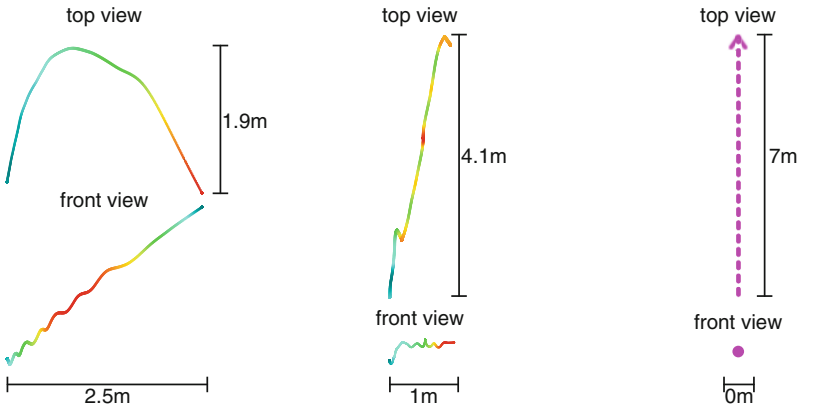

**(a)** Without pauses **(b)** With pauses **(c)** With pauses, without stabilization

**Fig. 15.** Three circles in a row with Samsung Galaxy S III

The last motion will be a walk. The author will hold the phone in his hand, will watch its display and will walk forward (i.e., in the direction of the $y$ axis)

through a seven metres long horizontal path. The result is depicted in Fig. 16a. The phone suffers from a strong gyroscope drift and the path is far from reality.

We will perform the same trick as with the circles and insert a small pause after each two steps. The result is depicted in Fig. 16b. The result has significantly improved, thanks to the stabilization system, and even individual steps leave a trace. Figure 16c shows the real path for comparison.




**(a)** Without pauses **(b)** With pauses **(c)** Expectation

**Fig. 16.** Walk - 7 m long - Samsung Galaxy S III

## 6 Conclusion and Future Work

In this paper, we presented a simple dead-reckoning method for tracking device movement and orientation using accelerometer, gyroscope, and magnetometer. A crucial role in our system plays a novel algorithm for correction of the supposed orientation whenever the device is at rest. We have evaluated the proposed methods in various experiments using two common smartphones.

We deliberately used a non-model method of tracking, which does not assume any knowledge about expected movement of the device. Exploiting some information about expected movements, for example using a Kalman filter, would undoubtedly be the way to further improve accuracy. Even without such information, minor improvements might be achieved by analyzing and eliminating sensor specific noise prior to the data fusion as discussed in Sect. 2.1.

Despite presence of the mentioned inertial sensors in modern electronics, these low-cost sensors are usually not accurate enough to be used for motion tracking. In this paper, we demonstrated that with some sensor data fusion we could track 3D motion with reasonable error for a couple of seconds, which might be useful for, e.g., gesture recognition. The proposed stabilization method can further extend this time, however, it requires the device to stop every few seconds. This requirement might be fulfilled, for instance, with shoe mounted

systems such as [15,16,22], where the stabilization algorithm could be reliably executed when the shoe touches the ground. The open question is whether similar stabilization can be implemented without the necessity to stop the device.

In spite of the fact that the inertial sensors do not provide sufficient tracking information by themselves, they can be very useful when fused with additional sensors. For instance, excelent results are demonstrated in the work by Mingyang Li *et al.* [11], where data from smartphone rolling-shutter camera were fused with data from inertial measurement unit using the extended Kalman filter. The authors were able to retain high accuracy even after ten minutes of walking.

**Acknowledgements.** Research was partially supported by the Czech Science Foundation under the project P103-15-19877S and by SVV under the project 260 224.

## References

1. Abbott, H., Powell, D.: Land-vehicle navigation using GPS. Proc. IEEE **87**(1), 145–162 (1999)
2. Barański, P., Bujacz, M., Strumillo, P.: Dead reckoning navigation: supplementing pedestrian GPS with an accelerometer-based pedometer and an electronic compass. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 7502, p. 16, June 2009
3. Chung, H., Ojeda, L., Borenstein, J.: Accurate mobile robot dead-reckoning with a precision-calibrated fiber-optic gyroscope. IEEE Trans. Robot. Autom. **17**(1), 80–84 (2001)
4. Fang, L., Antsaklis, P., Montestruque, L., McMickell, M., Lemmon, M., Sun, Y., Fang, H., Koutroulis, I., Haenggi, M., Xie, M., Xie, X.: Design of a wireless assisted pedestrian dead reckoning system - the NavMote experience. IEEE Trans. Instrum. Meas. **54**(6), 2342–2358 (2005)
5. Gusenbauer, D., Isert, C., Krosche, J.: Self-contained indoor positioning on off-the-shelf mobile devices. In: International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–9, September 2010
6. Jimenez, A., Seco, F., Prieto, J., Guevara, J.: Indoor pedestrian navigation using an INS/EKF framework for yaw drift reduction and a foot-mounted IMU. In: 7th Workshop on Positioning Navigation and Communication (WPNC), pp. 135–143, March 2010
7. Kalman, R.E.: A new approach to linear filtering and prediction problems. Trans. ASME-J. Basic Eng. **82**(Series D), 35–45 (1960)
8. Khosla, P., Khanna, R., Sood, S.: Quantification and mitigation of errors in the inertial measurements of distance. MAPAN **30**(1), 49–57 (2015). http://dx.doi.org/10.1007/s12647-014-0115-z
9. Kreyszig, E.: Advanced Engineering Mathematics, 8th edn. Wiley, New York (1999)
10. Lee, S.W., Mase, K.: Activity and location recognition using wearable sensors. IEEE Pervasive Comput. **1**(3), 24–32 (2002)
11. Li, M., Kim, B.H., Mourikis, A.: Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4712–4719, May 2013
12. Liu, M., Wang, H., Guo, Q., Jiang, X.: Research on particle filter based geomagnetic aided inertial navigation algorithm. In: 3rd International Symposium on Systems and Control in Aeronautics and Astronautics (ISSCAA), pp. 1023–1026, June 2010

13. Murray, G.: Rotation about an arbitrary axis in 3 dimensions (2013). http://inside.mines.edu/fs_home/gmurray/ArbitraryAxisRotation/ArbitraryAxisRotation.pdf. Accessed 24 September 2014
14. Neto, P., Norberto Pires, J., Moreira, A.: 3-D position estimation from inertial sensing: minimizing the error from the process of double integration of accelerations. In: 39th Annual Conference of the IEEE Industrial Electronics Society (IECON), pp. 4026–4031, November 2013
15. Ojeda, L., Borenstein, J.: Personal dead-reckoning system for GPS-denied environments. In: IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR), pp. 1–6, September 2007
16. Ruiz, A., Granja, F., Honorato, P.J., Rosas, J.: Accurate pedestrian indoor navigation by tightly coupling foot-mounted IMU and RFID measurements. IEEE Trans. Instrum. Meas. **61**(1), 178–189 (2012)
17. Shiau, J.K., Huang, C.X., Chang, M.Y., et al.: Noise characteristics of MEMS gyros null drift and temperature compensation. Appl. Sci. Eng **15**(3), 239–246 (2012)
18. Skog, I., Handel, P., Nilsson, J.O., Rantakokko, J.: Zero-velocity detection - an algorithm evaluation. IEEE Trans. Biomed. Eng. **57**(11), 2657–2666 (2010)
19. Stuart, R., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall, New York (2010)
20. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, Cambridge (2005)
21. Toledo-Moreo, R., Zamora-Izquierdo, M., Gomez-Skarmeta, A.: IMM-EKF based road vehicle navigation with low cost GPS/INS. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, pp. 433–438, September 2006
22. Weenk, D., Roetenberg, D., van Beijnum, B., Hermens, H., Veltink, P.: Ambulatory estimation of relative foot positions by fusing ultrasound and inertial sensor data. IEEE Trans. Neural Syst. Rehabil. Eng. **23**(5), 817–826 (2015)