Tutorial: Implementing a Pedestrian Tracker Using Inertial Sensors

Shoe-mounted inertial sensors offer a convenient way to track pedestrians in situations where other localization systems fail. This tutorial outlines a simple yet effective approach for implementing a reasonably accurate tracker.

Carl Fischer
Lancaster University, UK

Poorna Talkad Sukumar
Indian Institute of Science

Mike Hazas
Lancaster University, UK

Pedestrian dead-reckoning (PDR) using foot-mounted inertial measurement units (IMUs) is the basis for many indoor localization techniques, including map matching, various types of simultaneous localization and mapping (SLAM), and integration with GPS. Despite the increasing popularity of PDR methods over the past decade, there's little information about their implementation and the challenges encountered, even for very basic systems.

Some authors focus on the algorithmic details of PDR and use abstract formalisms, which can be daunting to readers who only require a simple implementation. Others assume that readers are familiar with PDR and focus on the additional sensors that distinguish their localization system from others. Here, we aim to make it easier for readers to use PDR as a component of a larger system by describing a standard inertial PDR method that's easy to implement. You can apply our method with minimal custom configuration, yet the method represents the foundation for state-of-the-art pedestrian inertial-tracking systems. We also reference key works that can provide a more formal understanding of the underlying principles and point to dedicated studies of particular aspects of pedestrian inertial tracking (see the sidebar). Furthermore, we give an honest account of the difficulties encountered when implementing, using, and evaluating a PDR system.

Inertial Pedestrian Dead-Reckoning

Dead reckoning is the process of estimating an object's position by tracking its movements relative to a known starting point. A typical example is a ship travelling at a constant speed in a fixed direction. The ship's current position is on a line starting at the known starting point in the direction of travel, and the distance from the starting point is given by the speed along the direction of travel multiplied by the time since the ship was at that known point. If the ship changes course, or if the speed changes, the navigator must note the current position estimate and start the process again using this new position estimate as the starting point for future estimates. Over time, these estimates become less accurate, because they rely on previous estimates, which are imperfect due to errors in speed and heading measurements. In other words, the small errors in heading and speed accumulate to form an increasingly large error in the position estimate.

The first pedestrian dead-reckoning methods applied exactly the same principles to estimate step-by-step positions, using a (digital) compass to measure the heading and an (electronic) pedometer to count steps. This method works in principle but assumes that the pedestrian is walking with steps of

Further Reading

Before trying to understand in detail the workings of the inertial pedestrian-tracking system we describe in this article, it is helpful to understand the basics of Kalman filtering and (non-pedestrian) inertial navigation.

Inertial Navigation and Kalman Filtering

An article by Dan Simon provides a good starting point for understanding the Kalman filter, using a simple vehicle tracking system as an example.[1] A technical report by Greg Welch and Gary Bishop is only slightly more formal and gives the fundamental Kalman *update* and *predict* equations.[2] It uses a simple example to illustrate the effects of adjusting the different filter parameters.

A textbook by David H. Titterton and John L. Weston lays the foundations of modern inertial navigation.[3] It rigorously defines the different reference frames and shows how Euler angles, rotation matrices, and especially quaternions can be used to represent the attitude (or orientation) of an inertial sensor. This book explains how an error-state Kalman filter, similar to the one we use, can fuse inertial navigation estimates with estimates from other navigation systems.

Another book on the topic of integrated navigation systems, by Paul D. Groves,[4] includes several chapters covering inertial navigation and Kalman filtering and includes additional topics such as filter behavior and parameter tuning. It describes different ways of integrating inertial tracking, as described in this tutorial, with magnetometers, altimeters, or GPS. These books provide two complementary views on a complex topic.

Pedestrian Dead-Reckoning Using Shoe-Mounted Inertial Sensors

Many research papers have examined pedestrian inertial navigation. We selected a few that we found helpful in designing the implementation suggested in this article. Lauro Ojeda and Johann Borenstein describe a system similar to our "naive implementation."[5] They designed it with emergency responders in mind, and tested it for various walking patterns, on stairs and on rugged terrain. Their simple algorithm performs well, thanks to the high-quality IMU they use, which is larger, heavier, and much more expensive than our MEMS sensors.

Raúl Feliz and his colleagues describe a similar system but with more emphasis on stance phase detection and velocity error correction.[6] Their system corrects position as well as velocity during zero-velocity updates (ZUPTs), using a less powerful but more intuitive alternative to the Kalman filter we describe in the main text. The 2005 article by Eric Foxlin is probably the most cited work in this area.[7] He explains clearly the benefits of using a Kalman filter to apply ZUPTs. Foxlin's article includes details of the implementation, but a more recent article by Antonio Jiménez and his colleagues gives a more complete description of the implementation process.[8] Their article also gives some tips for tuning a Kalman filter in the specific context of pedestrian inertial navigation.

REFERENCES

1. D. Simon, "Kalman Filtering," *Embedded Systems Programming*, June 1991, pp. 72–79.
2. G. Welch and G. Bishop, *An Introduction to the Kalman Filter*, tech. rep., University of North Carolina, Chapel Hill, 2006.
3. D.H. Titterton and J.L. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed., Institution of Engineering and Technology, 2004.
4. P.D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, Artech House, 2008.
5. L. Ojeda and J. Borenstein, "Non-GPS Navigation for Security Personnel and First Responders," *J. Navigation*, vol. 60, no. 9, 2007, pp. 391–407.
6. R. Feliz, E. Zalama, and J. G. García-Bermejo, "Pedestrian Tracking Using Inertial Sensors," *J. Physical Agents*, vol. 3, no. 1, 2009, pp. 35–43.
7. E. Foxlin, "Pedestrian Tracking with Shoe-Mounted Inertial Sensors," *IEEE Computer Graphics and Applications*, vol. 25, no. 6, 2005, pp. 38–46.
8. A. Jiménez et al., "Indoor Pedestrian Navigation Using an INS/EKF Framework for Yaw Drift Reduction and a Foot-Mounted IMU," *Proc. of Workshop on Positioning, Navigation and Communication* (WPNC 10), IEEE, 2010, pp. 135–143.

constant length. Cheap and small microelectromechanical (MEMS) accelerometers and gyroscopes have provided alternative methods. Typically, these are combined into an IMU comprising three accelerometers and three gyroscopes, aligned along three orthogonal axes. The accelerations are integrated to estimate velocity and position. Most of this method's complexity—and any errors—come from the fact that the accelerations are measured in the coordinate space attached to the IMU (the *sensor frame*), and not in a coordinate space easily associated with the room in which the experiment is taking place (the *navigation frame*). This is called a *strapdown* inertial navigation system, because the accelerometers rotate with the object being tracked.

Attitude and Heading Reference Systems

Many off-the-shelf IMUs include an attitude and heading reference system (AHRS) that estimates the transformation from the sensor frame to the navigation frame. In other words, the AHRS computes the orientation of the sensor in 3D space. It outputs the sensor's orientation as three Euler angles (roll, pitch, and yaw), or a 3×3 rotation matrix, or a quaternion. These are all equivalent ways of representing an orientation. We use the rotation matrix notation, which is the most intuitive.

An AHRS usually combines gyroscope readings with accelerometer readings, but sometimes it also combines gyro readings with accelerometer and magnetometer readings. The integrated gyroscope *rates of turn* (or speed of rotation) give the orientation. The accelerometers correct the sensor's tilt (roll and pitch), and the magnetometer corrects the sensor's heading (yaw). These corrections are necessary, because the orientation estimated from integrating the rates of turn accumulates error from the gyroscope noise. The accelerometer and magnetometer estimates of the orientation, on the other hand, don't accumulate error but are affected by the sensor's movement and magnetic interference, respectively. By combining these three sensor types, an AHRS can provide a good estimate of the sensor's orientation at all times. We have found that magnetometers are highly sensitive to unpredictable interference, so we deliberately omitted them in this tutorial.

The most straightforward implementations of inertial tracking use the orientations computed by an AHRS. However, many AHRSs use proprietary algorithms and are designed to work well for a range of applications. Few are designed specifically for foot-mounted pedestrian inertial tracking, where rates of turn and accelerations are much higher than those encountered in other fields. We found that by calculating the sensor's orientation in the inertial tracking algorithm itself, we can track the pedestrian more accurately and have more flexibility in tuning the algorithm's parameters.

Approximations and Assumptions

In pedestrian tracking, we use simplified inertial navigation equations for two reasons. First, distances and speeds are much smaller than for aircraft, ships, or land vehicles. Second, MEMS inertial sensors have relatively poor error characteristics when compared to the navigation-grade sensors typically used for vehicles. However, some of the constant bias and misalignment errors can be compensated for during a calibration phase, often performed by the manufacturer.

The full navigation equations compensate for various physical effects, which we can neglect with little consequence to the overall tracking error. For typical walking speeds, neglecting the centrifugal force due to the Earth's rotation causes a position error of 0.5 percent of the total distance travelled, if the experiment takes place at the equator, and less elsewhere. Neglecting this effect causes the estimated position to drift down and away from the equator by a small amount.

The Coriolis force—the effect by which the rotation of the Earth appears to deflect moving objects—is proportional to the target's speed relative to the Earth and to the Earth's rate of rotation. For a pedestrian, the effect is several orders of magnitude less than that caused by centrifugal force. In addition, some of the Coriolis errors will cancel out when the pedestrian changes direction, so there's not always accumulation of position error. The Earth's rotation is measured by the gyroscopes, but the Earth's rate of rotation (0.004 degrees per second) is far less than the bias drift (a slow but unpredictable error) of current MEMS gyroscopes (typically 0.1°/s) and can thus also be neglected.

If the tracked pedestrian remains within a few kilometers of his or her starting point, we can assume that the Earth's curvature over this area is negligible. This lets us work in a traditional Cartesian coordinate system, so we don't need to map the pedestrian's movements onto an ellipsoidal surface, which approximates the Earth's.

Compensating for all these errors would require an accurate estimate of the pedestrian's position and orientation relative to the Earth. This isn't possible using only inertial sensors; it would require additional technologies, such as GPS.

Zero-Velocity Detection

Zero-velocity (ZV) detection is an essential part of an inertial tracking system. Without this information, the velocity error would increase linearly with time, and the position estimate error would increase at least quadratically. ZV provides the required information to reset the velocity error.

Detecting when an inertial sensor is stationary can be challenging. During normal walking, ZV occurs during the *stance phase*—that is, when one foot is carrying the body's full weight. This has made foot-mounted IMUs a popular choice for pedestrian tracking. Tracking algorithms can use sensors mounted on other parts of the body to perform PDR by counting steps and estimating their length, but this isn't as accurate as methods using foot-mounted sensors with ZV every few seconds. John Elwell seems to be the first published researcher to have noted that each stance phase provides an opportunity to use ZV,[1] but an earlier unpublished project involving Larry Sher at DARPA also appears to have used similar techniques in 1996 (www.dist-systems.bbn.com/projects/PINS).[2]

There are several ways to detect ZV. One option is to use knowledge of the human walking pattern to detect the stance phase. Typically, such methods model walking as a repeating sequence of heel strike, stance, push off, and swing.[3] We expect these methods to fail for other modes of movement such as running, crawling, or walking backward.

A second, more generic option tries to determine when the sensor is stationary using only data from the inertial sensors. This option assumes that when the sensor is stationary, the measured acceleration is constant and equal to gravity, and the rates-of-turn measured by the gyroscopes are zero. Such methods can incorrectly detect ZV if the sensor moves at a constant speed, and they might fail to detect a stance phase if the sensors are very noisy. The occasional failure in ZV detection will increase the accumulated error in the position estimate but won't prevent the inertial tracking system from functioning.

If the estimated speed is incorrect, it will affect the estimated position in a predictable way.

Researchers have already compared different ZV or stance-phase detection methods, often examining the error on the total distance estimate or the error in the final position estimate,[4] but some have looked more closely at the number of steps detected.[5] For typical walking, the consensus seems to be that using the gyroscope rates of turn is the most reliable way of detecting ZV. However, Jonas Callmer and his colleagues suggest that including the accelerations in the detection provides better performance when the pedestrian is running.[5] In recent work, Özkan Bebek and his colleagues used high-resolution pressure sensors under the soles of a boot to detect when the foot is stationary.[6] They achieve slightly better tracking accuracy than when using gyroscopes because they can detect ZV more accurately. But, generally speaking, different stance-phase detection methods tend to have roughly equivalent performance.

Position Estimation

Our initial implementation used the orientation estimates from the IMU and simple velocity resets. We improved our results by using a Kalman filter to correct position and velocity estimates. Finally, we extended the Kalman filter to compute the orientation directly from the gyroscope and accelerometer measurements, thus giving us more control over our tracking system and better results.

Naïve implementation. The simplest implementation proceeds in five steps. First, transform the accelerations from the sensor frame into the navigation frame using the orientations estimated by the AHRS. Second, subtract gravity from the vertical axis. Third, integrate the accelerations to obtain the velocity. Fourth, reset the velocity to zero if the sensor is detected to be stationary. Finally, integrate the velocity to obtain the position.

Kalman filter implementation. We improve on naïve implementation by noting that velocity errors and position errors are correlated.[2] If the estimated speed is incorrect, it will affect the estimated position in a predictable way. In particular, when we detect that the sensor has stopped moving during a ZV phase, but the estimated speed isn't zero, we know that the position estimate will likely be incorrect. This means that whenever the sensor is detected to be stationary, we shouldn't just reset the estimated velocity to zero but should also adjust the estimated position by a small amount.

We separate the basic inertial navigation system (INS) from the ZV updates (ZUPTs). The INS transforms the accelerations into the navigation frame, subtracts gravity from the vertical axis, and integrates twice (integrating acceleration gives us velocity; integrating velocity gives us position) to obtain the velocity and position estimates. Because of the noisy accelerometer measurements, these basic INS estimates can be off by several meters after a few seconds. But, as in the previous method, we can reduce this error to acceptable levels by using ZUPTs.

The most common method used in the literature to implement this correction is the Kalman filter. A Kalman filter estimates the system state based on noisy measurements and a system model. In our tracking problem, the "system" is the INS (the foot, the sensor, and the simple integration algorithm); the "state" is the error in velocity and position estimates; and the "measurements" are the ZUPTs, which are virtual ZV measurements. In addition to the values of the velocity and position errors, the Kalman filter estimates their error covariances and cross-covariances. The cross-covariances let the filter correct the position (and not only the velocity) during a ZUPT. At the end of each ZUPT, the estimated errors in velocity and position are subtracted from the INS estimates.

This method is an error-state, or complementary, Kalman filter, and it's a common tool in multimodal navigation systems. The principles are the same as those used in a standard Kalman filter, but the implementation looks slightly different and, for our application, is simpler.

Estimating the IMU orientation. As mentioned earlier, commercial IMUs do an excellent job of estimating their orientation, but their AHRS algorithms are complex and usually inaccessible to users because of intellectual property issues. One problem we faced is that the AHRS embedded in our IMU performs some online calibration based on the sensor's type of movement. We noticed that the quality of orientation estimates (and therefore the accuracy of the tracking) often improves after a few minutes. This suggests that some internal parameters of the AHRS take time to reach their optimal values. By estimating the IMU's orientation ourselves, we can optimize the parameters for pedestrian motion and no longer depend on a proprietary algorithm. This lets us use other IMUs, but they might not compute orientation.

We estimate the orientation by integrating the rates of turn measured by the gyroscopes. The estimated orientations inevitably suffer from drift, but the algorithm corrects them during the ZUPTs. In the same way that the position estimates are correlated with the speed, so is the orientation. It can therefore be corrected by our Kalman filter, even though it's not measured directly.

Intuitively, if the orientation is incorrect, the gravity component won't be entirely removed from the acceleration. The remaining gravity component will be integrated, causing a velocity error that's correlated with the orientation error. Thus, there's a strong correlation between the tilt errors (roll and pitch) and the velocity errors, because gravity and foot impact occur on the vertical axis. There's less correlation between yaw (or heading) error and velocity, and thus less correction of the yaw. To minimize yaw drift, we compensate for as much of the constant gyroscope bias as possible by averaging measurements during a stationary period before each run.

The main difference between this method and the previous Kalman filter method is that the orientation must be estimated using the gyroscope readings. Also, the orientation error appears in the state vector along with the velocity and position errors. Figure 1 describes this algorithm.

Practical Implementation

Here, we give a step-by-step guide to implementing a pedestrian inertial tracking system that uses an error-state Kalman filter for ZUPTs and orientation estimation. Although this is a more complex implementation than the straightforward naïve implementation we described first, it takes less than 60 lines of Matlab code and produces much better results. The Matlab implementation and a few sample recordings are available at http://doi.ieeecomputersociety.org/10.1109/MPRV.2012.16.

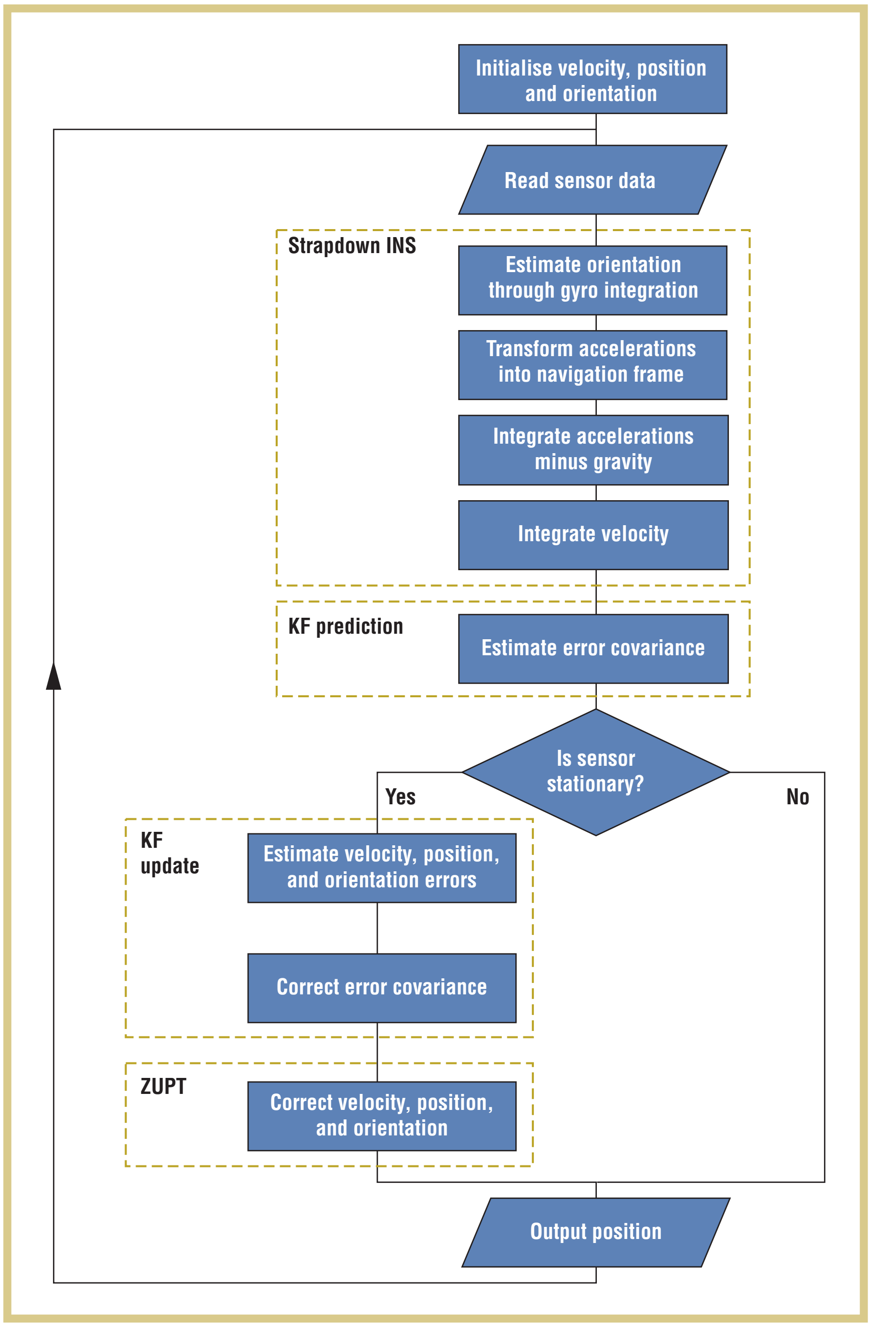

Initialise velocity, position and orientation
Read sensor data
Strapdown INS
Estimate orientation through gyro integration
Transform accelerations into navigation frame
Integrate accelerations minus gravity
Integrate velocity
KF prediction
Estimate error covariance
Is sensor stationary?
Yes
No
KF update
Estimate velocity, position, and orientation errors
Correct error covariance
ZUPT
Correct velocity, position, and orientation
Output position

Figure 1. An inertial navigation system with orientation estimation corrected by Kalman filter zero-velocity updates (ZUPTs).

Initialization

Before the main data processing loop, we define the parameters (see Table 1) and declare and initialize the variables. This initialization is valid if the sensor is stationary. If the sensor is moving, the first few steps will be incorrectly estimated until the system adapts (see Figure 2).

Zero-Velocity Detection

Next, we apply the ZV test to each data sample within the main loop. We achieved good results using the detection method described by Antonio R. Jiménez and his colleagues.[7] They applied thresholds to the acceleration magnitude, gyroscope magnitude, and

TABLE 1
System parameter values.

Name	Notation	Value
Time step	Δt	1/120 second
Accelerometer noise	σ_a	0.01 m/s^2
Gyroscope noise	σ_ω	0.01 rad/s (radian per second)
Gravity	g	9.8 m/s^2
Zero-velocity (ZV) measurement noise	σ_v	0.01 m/s
Gyroscope ZV detection threshold	σ_ω	0.6 rad/s

- Define the zero-velocity (ZV) measurement matrix: $\mathbf{H} = (0_{3\times3}\ 0_{3\times3}\ \mathbf{I}_{3\times3})$
- Define the ZV measurement noise covariance $\mathbf{R}$ as the diagonal matrix with values $\begin{pmatrix} \sigma_{v_x}^2 & \sigma_{v_y}^2 & \sigma_{v_z}^2 \end{pmatrix}$
- Initialize the position: $\mathbf{p} = (0\ 0\ 0)^T$
- Initialize the velocity: $\mathbf{v} = (0\ 0\ 0)^T$
- Initialize the error covariance matrix: $\mathbf{P} = 0_{9\times9}$
- Initialize the orientation matrix $\mathbf{C}$ based on the accelerations:

$$roll = \arctan(a_y^{sensor} / a_z^{sensor})$$
$$pitch = -\arcsin(a_x^{sensor} / g)$$
$$yaw = 0$$
$$\mathbf{C} = \begin{pmatrix} \cos(pitch) & \sin(roll)\sin(pitch) & \cos(roll)\sin(pitch) \\ 0 & \cos(roll) & -\sin(roll) \\ -\sin(pitch) & \sin(roll)\cos(pitch) & \cos(roll)\cos(pitch) \end{pmatrix}$$

Figure 2. Before the main data processing loop, we define the parameters and declare and initialize the variables. Table 1 presents the parameter values.

local acceleration variance. However, a simple threshold on the magnitude of the gyroscope rate-of-turn measurements also works well. A ZUPT is applied when $\|\omega\| < \alpha_\omega$. Isaac Skog and his colleagues confirmed this, showing that their gyroscope-based detector works as well as the one using both accelerometers and gyroscopes for typical walking.[8]

Main Loop

The operations in Figure 3 are performed for every measurement sample. The sample index, k, is occasionally omitted to simplify notations. For an explanation of the orientation matrix update in steps 4 and 17 in Figure 3, we refer you to work by Honghui Qi and J.B. Moore.[9] Dan Simon provides more details regarding the Joseph form of the covariance update used in step 15.[10] The sidebar provides more general references, which cover the general Kalman filter and inertial navigation equations.

Evaluation

We don't aim to give a precise comparison of different PDR techniques or to determine optimal parameters. Rather, we want to provide a practical guide, outlining the level of performance you can expect from the standard implementation we've described.

Detailed ground truth is difficult to obtain, but even when it's available, the challenges of aligning it with the estimated path and computing position errors remain. All position estimates are relative to the initial position and heading, and a small error early in the path can have a significant effect later on, even if no further errors occur.

Researchers have used several methods to evaluate the performance of PDR systems, including

- inspecting estimated paths in the coordinate space and comparing them to a simple geometric ground truth;
- computing the distance between starting point and final position estimate for a closed loop walk—smaller distances indicate less drift (but this doesn't account for scaling error);
- computing the estimated total distance travelled, which will give different results depending on whether distance is accumulated per measurement or per step (and this doesn't account for heading errors); and
- computing the distance and heading error for each step (or segment) by realigning each previous step (or segment) to the ground truth, as Michael Angermann and his colleagues suggest.[11]

To evaluate the quality of ZV detection, we compare ZV detection using the IMU to ZV detection using force-sensitive resistors on the sole of the footwear or video recordings of foot movement.

We tested our pedestrian-tracking system with six people. Each subject walked approximately 240 meters in our office building, including several flights of stairs, in approximately four minutes. We used an XSens MTx sensor (model MTx-28A53G25 or MTx-49A53G25) to record inertial measurements at 120 samples per second. We use the sensor's *calibrated* output for precision. These sensors are the standard model with accelerometers with a full scale of ± 50 m/s^2 and gyroscopes with a full scale of ±1,200 °/s. We have since realized that the accelerometer range is insufficient for optimal tracking; the ± 180 m/s^2 model would have been more suitable. The sensor was attached to the instep of the foot with a Velcro strap.

1. Compute the time step Δt from the previous measurement. Typically, Δt is constant.
2. Subtract the gyroscope bias from the measurements if required.
3. Compute the skew-symmetric angular rate matrix Ω from the gyroscope readings:

$$\Omega_k = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}.$$

4. Update the orientation matrix: $\mathbf{C}_k = \mathbf{C}_{k-1} = (2\mathbf{I}_{3\times 3} + \Omega_k \Delta t)(2\mathbf{I}_{3\times 3} - \Omega_k \Delta t)^{-1}$.[9] Post-multiply by the update factor, because the gyroscope measurements are taken in the sensor frame.
5. Transform the measured accelerations from the sensor frame into the navigation frame: $\mathbf{a}_k^{nav} = (\mathbf{C}_k + \mathbf{C}_{k-1})\,\mathbf{a}_k^{sensor}/2$. Use the average of the previous and the current orientation estimate, because movement has occurred between measurements (as a rough approximation).
6. Integrate the acceleration in the navigation frame minus gravity to obtain the velocity estimate: $\mathbf{v}_k = \mathbf{v}_{k-1} + \left(\mathbf{a}_k^{nav} + \mathbf{a}_{k-1}^{nav} - 2\begin{pmatrix} 0 & 0 & g \end{pmatrix}^T\right)\Delta t/2$. Use the trapeze method of integration.
7. Integrate the velocity to obtain the position estimate: $\mathbf{p}_k = \mathbf{p}_{k-1} + (\mathbf{v}_k + \mathbf{v}_{k-1})\Delta t/2$.
8. Construct the skew-symmetric cross-product operator matrix $\mathbf{S}$ from the navigation frame accelerations:

$$\mathbf{S}_k = \begin{pmatrix} 0 & -a_z^{nav} & a_y^{nav} \\ a_z^{nav} & 0 & -a_x^{nav} \\ -a_y^{nav} & a_x^{nav} & 0 \end{pmatrix}.$$

   This matrix relates the variation in velocity errors to the variation in orientation errors.
9. Construct the state transition matrix:

$$\mathbf{F}_k = \begin{pmatrix} \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{I}_{3\times 3} & \mathbf{I}_{3\times 3}\Delta t \\ -\mathbf{S}_k\Delta t & \mathbf{0}_{3\times 3} & \mathbf{I}_{3\times 3} \end{pmatrix}.$$

10. Construct the process noise covariance matrix $\mathbf{Q}_k$ as the diagonal matrix with values $\left(\begin{pmatrix} \sigma_{\omega_x} & \sigma_{\omega_y} & \sigma_{\omega_z} & 0 & 0 & 0 & \sigma_{a_x} & \sigma_{a_y} & \sigma_{a_z} \end{pmatrix}\Delta t\right)^2$.
11. Propagate the error covariance matrix $\mathbf{P}_k = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k$. This is valid if we assume process noise is identical on all axes.
12. Detect a stationary phase.

The following operations are only performed for samples occurring in a stationary phase.

13. Compute the Kalman gain $\mathbf{K}_k = \mathbf{P}_k \mathbf{H}^T (\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$.
14. Compute state errors from the Kalman gain and estimated velocity: $\varepsilon_k = (\varepsilon_C\, \varepsilon_P\, \varepsilon_v)^T = \mathbf{K}_k \mathbf{v}_k$. Here, the complete error vector ε is composed of the three elements of the attitude error (error on roll, pitch, and yaw angles), the position error, and the velocity error, in that order.
15. Correct the error covariance: $\mathbf{P}_k = (\mathbf{I}_{9\times 9} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k$. Alternatively, use the more robust Joseph form—see equation 5.19 in Dan Simon's *Optimal State Estimation.*[10]
16. Construct the skew-symmetric correction matrix for small angles:

$$\Omega_{\varepsilon,k} = \begin{pmatrix} 0 & \varepsilon_C[3] & -\varepsilon_C[2] \\ -\varepsilon_C[3] & 0 & \varepsilon_C[1] \\ \varepsilon_C[2] & -\varepsilon_C[1] & 0 \end{pmatrix}.$$

   The indices are one-based, so $\varepsilon_C[1]$ is the first element of the attitude error (roll).
17. Correct the attitude estimate: $\mathbf{C}_k = (2\mathbf{I}_{3\times 3} + \Omega_{\varepsilon,k})(2\mathbf{I}_{3\times 3} - \Omega_{\varepsilon,k})^{-1}\mathbf{C}_k$.[9] Pre-multiply by the correction factor, because the correction is computed in the navigation frame.
18. Correct the position estimate: $\mathbf{P}_k = \mathbf{P}_k - \varepsilon_p$.
19. Correct the velocity estimate: $\mathbf{v}_k = \mathbf{v}_k - \varepsilon_v$.

Figure 3. Operations performed for every measurement sample. The sample index, *k*, is occasionally omitted to simplify notations.

Figure 4 shows the horizontal and altitude plots from our three implementations: the naive implementation and the Kalman filter implementation (both using the orientations estimated by the built-in AHRS), and the Kalman filter implementation estimating the orientation itself. The fourth pair of plots (Figure 4d) shows the improvement in horizontal position estimates when we manually correct the gyroscope bias. The plots chosen for this article illustrate the improvement achieved for the altitude estimates using the Kalman filter for the ZUPTs, and the additional improvement in horizontal position estimates when we compute the orientations ourselves and compensate for gyroscope bias.

The difference between subjects and sensors was significant. For some subjects, we couldn't achieve accurate altitude estimates (there was a drift of up to four meters over the whole walk). On the other hand, some datasets didn't require any gyroscope bias correction and produced accurate horizontal plots, even with the simpler implementations. Nevertheless, the Kalman filter implementation with orientation estimation and manual gyroscope bias correction always produced the best results. The stairs are clearly visible at both ends of the building, as well as two detours to avoid seating areas, and even a small kink in the middle of a corridor where the subject paused to open a door. Data recorded at 120 samples per second gave the best results; lower sample rates displayed degraded performance, but higher rates didn't bring any noticeable improvement.

We achieved good results with the pedestrian inertial data provided online by the team at the German Aerospace Centre.[11] We also tested our implementation for running. The results were reasonable but would improve considerably if the accelerometers had a wider measurement range to accurately record the foot impacts.

Challenges and Suggestions

Two years ago, we started investigating pedestrian inertial tracking as a component of a larger multimodal indoor localization system. Since then, we've moved from the naïve implementation described earlier to the Kalman filter implementation. More recently, we've started estimating the orientation in the algorithm itself. Each new implementation has brought a better understanding of the inner workings of pedestrian inertial tracking.

Major Causes of Tracking Errors

The gyroscope bias caused most of the horizontal errors observed during our work. We now recalibrate our gyroscopes before each experiment by recording data for 30 to 60 seconds while the sensor isn't moving. The mean of the gyroscope readings for each axis gives us the bias, which must be subtracted from all measurements before running the PDR algorithm. This method works well even if the sensor is attached to a pedestrian's foot during calibration. This is a sensor-specific calibration that should be performed again as the components age, or if they're subject to a violent shock. Although others have included gyroscope and accelerometer bias estimation in their pedestrian inertial navigation systems, such methods don't appear to improve position estimates in the context of pedestrian tracking.

Altitude errors are also common. W. Todd Faulkner and his colleagues have analyzed the altitude errors in inertial pedestrian-tracking systems.[12] They found that such errors are due to the accelerations of the foot exceeding the dynamic range of the accelerometers. They noted that accelerations can reach ±10 *g* when walking, and ±13 *g* when descending stairs or running. This confirms our own observations and explains why the quality of our altitude estimates varies so much between users with our 5 *g* accelerometers. For the best results, developers should use accelerometers with at least a ±10 *g* range and gyroscopes with a ±900 °/s range. Currently, these are relatively high specifications for MEMS inertial sensors.

Parameter Tuning

It's reasonable to tune the system parameters to some extent based on sensor characteristics and prior knowledge of how the system will be used. However, the system should perform as well as possible for various types of movement and without any user-specific training. The choice of one parameter value will affect some of the others. In some cases, a poor choice of one parameter or even an error in the implementation can mask another incorrect parameter. Table 1 lists the parameters used in our final implementation. You should get acceptable results with these values, but you might be able to improve the results by further tuning these parameters. John-Olof Nilsson and his colleagues give a more formal analysis of how different parameters affect performance.[13] Note that due to an error, the values they give for the noise parameters should be divided by 250 (the sampling frequency).

Step 10 (in Figure 3), which estimates the process noise **Q**, simply multiplies the gyroscope and accelerometer noise (σ_ω and σ_a) by the timestep to determine the orientation and velocity noise, and sets position noise to zero. Note that these noise values account for all sources of error in the INS—not only short-term sensor noise but also bias variations, scaling errors, and integration errors. They're therefore different from the noise values reported in sensor datasheets.

The ZV measurement noise σ_v, used to construct **R**, represents the uncertainty in velocity during a ZUPT and should thus be one or two orders of magnitude smaller than an average walking speed of 2 m/s. A more precise ZV detector would require a smaller value, and a less precise one, a larger value. Generally, if the ratio of **Q**/**R** remains constant, the system will

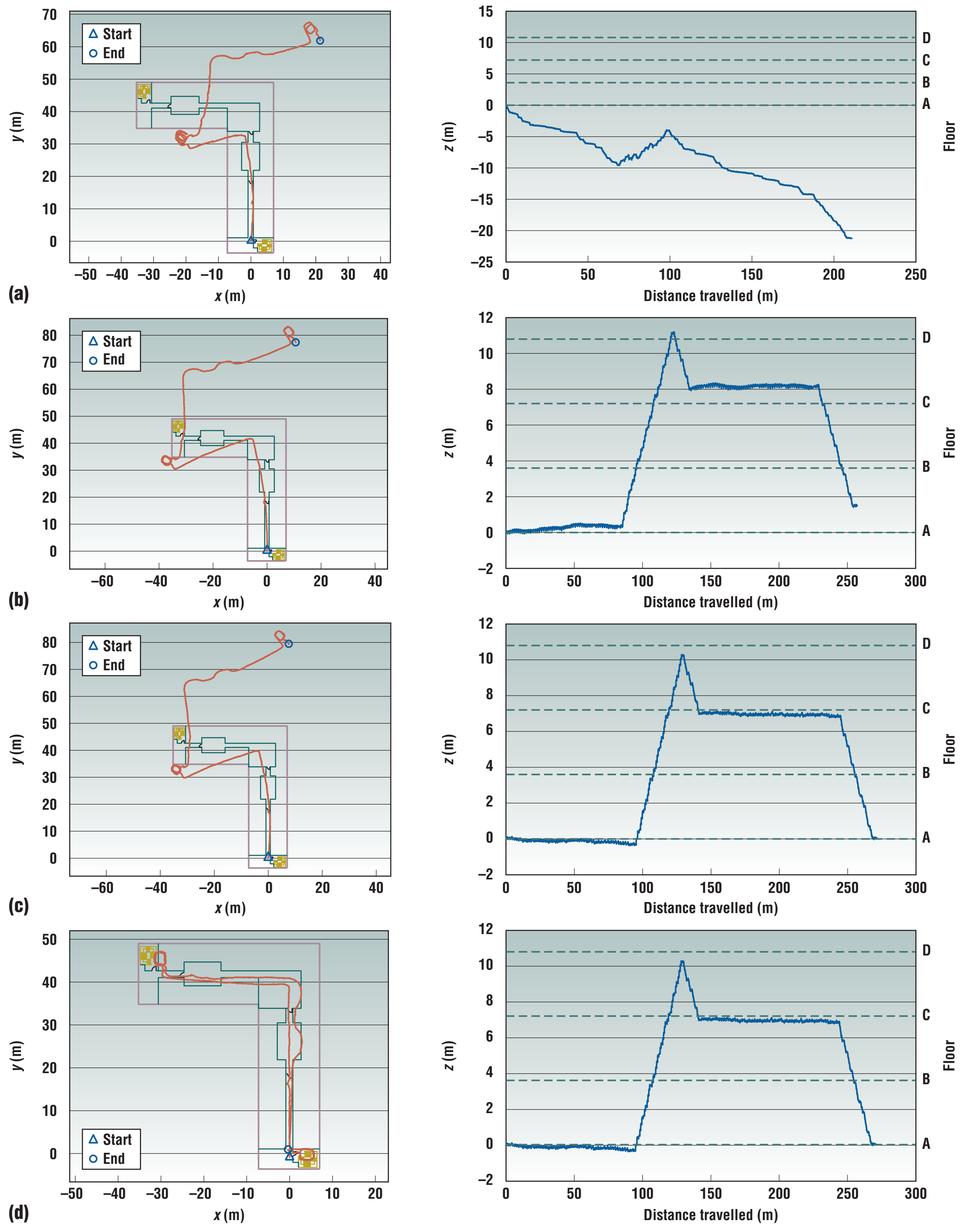
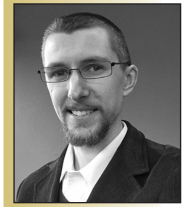


Figure 4. Plots for a four-minute walk through the Infolab, including stairs at both ends of the building. Each pair of plots shows the horizontal position (left) and altitude (right) for (a) naïve ZUPT, (b) KF ZUPT, (c) KF with orientation estimation, and (d) KF with orientation estimation and gyroscope bias corrections.

the AUTHORS



Carl Fischer is a senior research associate in the School of Computing and Communications at Lancaster University, UK. His research interests include pedestrian navigation in uninstrumented environments and robust positioning systems. Fischer received his PhD in computing from Lancaster University. Contact him at c.fischer@lancaster.ac.uk.

Poorna Talkad Sukumar is a project associate in the Computer Science and Automation department at the Indian Institute of Science, Bangalore, India. Her research interests include sensor systems, positioning and navigation, computer vision-based UI, and activity recognition. Talkad Sukumar received her MSc in mobile and ubiquitous computing from Lancaster University. Contact her at poorna.t.s@gmail.com.

Mike Hazas is an academic fellow and lecturer in the School of Computing and Communications at Lancaster University, UK. His research interests include energy and sustainability in everyday life, real world sensing, and algorithms. Hazas received his PhD in computing from the University of Cambridge. Contact him at hazas@comp.lancs.ac.uk.

perform the same. We recommend setting **R** and then adjusting **Q**.

We found an approximate value for the gyroscope ZV detection threshold α_ω by plotting the norm of gyroscope rates of turn over time for a sample pedestrian recording. Periodic stance phases are easy to recognize in such a plot. We set the threshold value slightly higher than the value of the norm during these phases and then adjusted it to get the best tracking results on the sample recording. This value worked well for all our recordings, but an adaptive threshold could provide a more robust solution.

Output Format

The output of a PDR system can take several equivalent forms, which should be chosen according to the application. If the system will be used alone, it can simply output the estimated Cartesian coordinates of the pedestrian. If the PDR system is just one component of a larger localization system, it might be more convenient to output incremental values, such as step length, variation in travel direction, and variation in altitude. Some applications might not require or be able to process position updates at the high sampling rate of the inertial sensors. In this case, the PDR system should output estimates only once per step, or at a fixed rate.

Tracking the Orientation

In some applications, tracking the yaw of the IMU as well as the direction of travel might be useful. This lets us distinguish between the pedestrian walking forward or backward, which is particularly useful in applications where we want to orient a display according to the direction the user is facing (rather than the direction in which the user is walking). Assuming the x-axis of the IMU is aligned with the pedestrian's *forward* direction, we can easily compute the yaw from the orientation matrix as $yaw - arctan(C_{2,1}/C_{1,1})$.

A better way is to use a second IMU attached to the pedestrian's chest or head and an associated Kalman filter to track the IMU's position. ZUPTs aren't possible, but we can assume a constant distance between the two IMUs and insert virtual position measurements.

Common Mistakes

During our development of PDR systems, we committed several trivial errors. Developers shouldn't confuse the orientation matrix with its transpose; this will give meaningless position estimates. When updating or correcting the orientation matrix, developers must take care to post-multiply or pre-multiply as appropriate. Notations and definitions vary between published works, and we have noted small errors that completely change an equation's meaning.

We also committed a more fundamental error by subtracting the acceleration due to gravity from the measured acceleration immediately after transforming it into the navigation frame. The acceleration due to gravity should indeed be removed during the integration phase. However, it should be left untouched when constructing the skew-symmetric matrix **S** to preserve the correlation between orientation and velocity.

More recently, we've used this tracking system as the basis for a navigation system for firefighters, which we built using SLAM. So far, we've evaluated the navigation system using a video game, but we believe real-world tests will yield promising results. P

ACKNOWLEDGMENTS

Many thanks to John-Olof Nilsson at KTH for his invaluable insights into parameter tuning, and to him and others for independently testing our code with their data.

REFERENCES

1. J. Elwell, "Inertial Navigation for the Urban Warrior," *Proc. of Digitization of the Battlespace IV*, SPIE, vol. 3709, 1999, pp. 196–204.

2. E. Foxlin, "Pedestrian Tracking with Shoe-Mounted Inertial Sensors," *IEEE*

Computer Graphics and Applications, vol. 25, no. 6, 2005, pp. 38–46.

3. S.K. Park and Y.S. Suh, "A Zero Velocity Detection Algorithm Using Inertial Sensors for Pedestrian Navigation Systems," *Sensors*, vol. 10, no. 10, 2010, pp. 9163–9178.

4. I. Skog, J.-O. Nilsson, and P. Händel, "Evaluation of Zero-Velocity Detectors for Foot-Mounted Inertial Navigation Systems," *Proc. 2010 Int'l Conf. Indoor Positioning and Indoor Navigation*, extended abstract, IEEE, 2010, pp. 172–173.

5. J. Callmer, D. Törnqvist, and F. Gustafsson, "Probabilistic Stand Still Detection Using Foot Mounted IMU," *Proc. 13th Int'l Conf. Information Fusion*, IEEE, 2010, pp. 1–7.

6. Ö. Bebek et al., "Personal Navigation via High-Resolution Gait-Corrected Inertial Measurement Units," *IEEE Trans. Instrumentation and Measurement*, vol. 59, no. 11, 2010, pp. 3018–3027.

7. A. Jiménez et al., "Indoor Pedestrian Navigation Using an INS/EKF Framework for Yaw Drift Reduction and a Foot-Mounted IMU," *Proc. of Workshop on Positioning, Navigation and Communication* (WPNC 10), IEEE, 2001, pp. 135–143.

8. I. Skog et al., "Zero Velocity Detection—An Algorithm Evaluation," *IEEE Trans. Biomedical Eng.*, vol. 57, no. 11, 2010, pp. 2657–2666.

9. H. Qi and J. B. Moore, "Direct Kalman Filtering Approach for GPS/INS Integration," *IEEE Trans. Aerospace and Electronic Systems*, vol. 38, no. 2, 2002, pp. 687–693.

10. D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, John Wiley, 2006.

11. M. Angermann et al., "A High Precision Reference Data Set for Pedestrian Navigation Using Foot-Mounted Inertial Sensors," *Proc. 2010 Int'l Conf. Indoor Positioning and Indoor Navigation* (IPIN 10), IEEE, 2010, pp. 1–6.

12. W.T. Faulkner et al., "Altitude Accuracy While Tracking Pedestrians Using a Boot-Mounted IMU," *Position Location and Navigation Symposium* (PLANS 10), IEEE, 2010, pp. 90–96.

13. J.-O. Nilsson, I. Skog, and P. Händel, "Performance Characterisation of Foot-Mounted ZUPT-Aided INSs and Other Related Systems," *Proc. 2010 Int'l Conf. Indoor Positioning and Indoor Navigation*, extended abstract, IEEE, 2010, pp. 182–183.