Acrux: Indoor Localization Without Strings

Jean-Gabriel Krieg  
IRIT-INPT/ENSEEIHIT  
University of Toulouse  
Toulouse, France  
jeangabriel.krieg@enseeih.fr

Hadrien Toma  
ENSICAEN  
Caen, France  
hadrien.toma@ecole.ensicaen.fr

Gentian Jakllari  
IRIT-INPT/ENSEEIHIT  
University of Toulouse  
Toulouse, France  
jakllari@enseeih.fr

Andre-Luc Beylot  
IRIT-INPT/ENSEEIHIT  
University of Toulouse  
Toulouse, France  
beylot@enseeih.fr

ABSTRACT
We present Acrux, the first indoor localization system to achieve meter level accuracy while relying exclusively on a single fix and the sensors commonly found in off-the-shelf smartphones. Acrux uses dead-reckoning, the approach that gives probably the best chance at a completely autonomous indoor localization system. Unfortunately, it has not been mastered on smartphones beyond a few dozen meters due to its inherent integration drift. As a result, all dead-reckoning based solutions in literature require periodic recalibration using input from outside – attaching strings preventing indoor localization from becoming mainstream. While it is virtually impossible to completely eliminate integration drift, Acrux is the first solution to succeed in dead-reckoning with meter level accuracy for several hundred meters, enough to relax the requirement for periodic recalibration in most indoor scenarios. To accomplish this, Acrux replaces step-counting, the standard approach for measuring distance using sensors, with an approach that measures the speed of locomotion. Although a straightforward accurate estimation of motion speed using the erroneous sensors found on smartphones is infeasible, Acrux combines a novel approach with measurement based analysis to achieve that. Leveraging its excellent dead-reckoning capability, Acrux is shown to provide indoor localization with median error between 0.7 m and 1.2 m and 98% percentile error of 3 m in a dozen of scenarios in 4 different buildings – without any recalibration.

KEYWORDS
Indoor localization; Dead reckoning; Inertial navigation system

1 INTRODUCTION
Someone named Alice, smartphone in her hand, enters an area with weak or no satellite coverage – inside a building, subway station, urban canyon – and losses the GPS fix. An app on the phone picks up where the GPS left off and continues to provide accurate fixes to help Alice get to her destination. The app is standalone and autonomous – it relies on the smartphone’s sensors only. While this scenario might sound well within our technological capabilities, it is currently impossible.

Extending the localization experience enabled by GPS in most outdoors areas to areas with weak or no satellite reception – the last-mile localization problem – has been the subject of intense research and development in recent years. A majority of the proposed solutions rely on the smartphone’s Wi-Fi transceiver and available Wi-Fi infrastructure [5, 14, 19, 26]. However, a training phase, known as fingerprinting, is required for every environment – an onerous task that has to be repeated every time the Wi-Fi infrastructure is updated. Crowdsourcing has been proposed [5, 8, 23] to remove the pain from fingerprinting. However, crowd-based approaches suffer from a chicken-egg problem: early adaptation hinges on the quality of the solution while the quality of the solution hinges on a high number of adapters. Other RF-based solutions relax the requirement for fingerprinting but require specialized hardware [28] and/or the deployment of dedicated servers [16, 17], creating a high barrier to entry. With the increase in sophistication of the smartphone hardware many responded by designing indoor localization systems that leverage the motion and/or audio/video sensors. In principle, equipped with inertial sensors, Alice should be able to track her trajectory once outside the GPS’ reach and using the last GPS fix continue generating accurate fixes until reaching her destination. This approach, commonly known as dead-reckoning, has been in use since the days of Christopher Columbus for determining longitude [7] but it suffers from integration drift [10]. No solution currently exists that can make it work for more than a few dozen meters on a commodity smartphone [5, 6, 18]. As a result, the common approach is to perform periodic recalibration every few dozen meters: using the Wi-Fi infrastructure [5], an accurate geometry of the location [6, 18] and/or crowdsourcing [23].

We present Acrux, the first system that relies exclusively on the smartphone’s sensors and a single fix to address the last-mile localization problem. Acrux achieves this by improving the accuracy of dead-reckoning by several times when compared to state of the art solutions thanks to two innovations. First, we make an
observation that becomes the foundation of this work’s novel contribution. The natural language contains several nonsynonymous terms for describing human motion indoors: strolling, rushing, walking, running. Obviously, this is because we, as humans, have noticed differences in the body mechanics and speed of locomotion. Yet, the dead-reckoning schemes proposed so far consider all human motion indoors as simply “walking” and try to estimate the distance a user has “walked” since the last fix. A system that would first identify if a user is strolling, walking, rushing or running – which we collectively refer to as elementary locomotions – and then estimate the distance she could travel given the elementary locomotion would benefit from higher granularity and thus have the potential to be significantly more accurate. Second, instead of following the common approach of estimating the distance traveled by counting steps and multiplying by an estimate of step size, something notoriously difficult to measure accurately[18, 27], AcruX estimates the speed of movement. Speed is less variable – people walking together may have very different step sizes but they progress at similar speeds.

Translating these observations into an accurate and efficient localization system running entirely on a commodity smartphone entails several challenges: (i) While humans can easily distinguish the different elementary locomotions, how can AcruX do so relying purely on the inertial sensors that are erroneous and are further affected by how/where users carry their phones; (2) Even if AcruX is capable of recognizing with accuracy the different elementary locomotions when given enough sensor readings, how will it be able to determine each elementary locomotion in real-time and on a smartphone with limited processing power running on battery; (3) How does AcruX use this information to compute speed and direction so as to fully map a user’s trajectory; (4) What happens when a user moves between floors.

In short, we address these challenges by first identifying unique patterns in the inertial sensor readings, vertical acceleration in particular, that unequivocally identify each elementary locomotion, regardless of the particular user’s height and weight. Using the identified patterns, we design and implement an algorithm that can recognize an elementary locomotion on a commodity smartphone using only 1 s of sensor readings. This information is combined with the user’s cadence for accurately estimating the speed of locomotion. To estimate the direction of locomotion, AcruX combines the knowledge of the current elementary locomotion with an approach for correcting bias into a light-weight and accurate algorithm. Finally, combining vertical acceleration, barometer readings and knowledge of regulations regarding the angle of inclination of stairs and escalators, AcruX is able to accurately track a user as she moves between floors.

Our main contributions may be summarized as follows:

- We extend AcruX in “3D” (§ 5), enabling it to provide localization fixes when a user moves between floors.
- We develop AcruX on the Android OS and carry out extensive experiments on 4 commodity smartphones and 4 different indoor locations, including two subway stations (§ 6). Measurements on a 5 km path show that the best step-counting strategy deviates from the truth by 1 m after 99 m and by 55.29 m at the end of the path; AcruX deviates by 1 m after 557 m and by 8.33 m at the end of the 5 km path – an improvement of over 5.6%. Dozens of scenarios in 4 different buildings showed median errors between 0.7 and 1.2 m and 98% percentile error of 3 m.

2 OVERVIEW OF ACRUX

Fig. 1 shows a high-level depiction of AcruX’s architecture. The heart of AcruX is utilizing the readings from the accelerometer, gyroscope and barometer for computing the smartphone user’s motion velocity. When AcruX determines the phone has lost the GPS lock, it does two things. First, it stores a fix consisting of the coordinates and global direction. Second, it establishes its own frame of reference. Once inside the building, all localization computations are done in its own frame of reference and converted into global coordinates using the initial fix. As the raw gyroscope and accelerometer readings are in the phone’s frame of reference, the Phone Attitude component converts the readings into the user’s frame of reference before passing them onto the rest of AcruX.

There are three key components that utilize the sensors readings for accurately computing velocity: the Speed Estimation component (§ 3), the Direction Estimation component (§ 4) and what we refer to as 3D (§ 5), the component for tracking a user as she transitions between floors. This includes a user taking the stairs, escalator (standing or walking) as well as the elevator. In the following, we describe each component in detail.

3 ESTIMATING SPEED

In this section, we describe how AcruX estimates the speed of locomotion, one of the two components required for computing the velocity. AcruX uses a two step algorithm that first limits the range of possible speed values and then guesses the actual speed. To limit the range of possible speed values, AcruX deconstructs human motion into elementary locomotions and then solves the challenge of identifying these locomotions in real time while relying exclusively on inertial sensors. AcruX’s solution requires a single signature per

\[ \text{footnote}{2}\text{This information can be extracted from the phone’s operating system. Android, for example, exposes a very detailed state of the GPS.} \]

\[ \text{footnote}{3}\text{There are more sophisticated systems for accurately computing the phone’s attitude [30]. However, as AcruX relies on patterns in the sensor readings rather than their absolute values, a heavier, albeit more accurate, system is not necessary.} \]
elementary locomotion and only 1 s of sensor readings; no per user training is required. Once the elementary locomotion is identified, Acrux measures the user’s current cadence and uses it to estimate the speed of locomotion. In the following, we describe both steps in detail.

3.1 Identifying the Elementary Locomotions

Inspired by how humans perceive pedestrian motion indoors, reflected in the nonsynonymous terms of the natural language, Acrux deconstructs human motion into 6 elementary locomotions: (1) Static – no movement; (2) Idle: random movement with no localization change, e.g. while sitting in a food court at the mall; (3) Strolling; (4) Walking; (5) Rushing and (6) Running.

We leverage the sensor readings in response to the distinct body mechanics inherent in every elementary locomotion to build a signature for every elementary locomotion. Acrux stores the signature and when requested it can identify the user’s current elementary locomotion by comparing the phone’s sensor readings to the stored signatures. The challenge, however, is building a single signature per elementary locomotion, regardless of a user’s height, weight or gender.

3.1.1 Building a Signature per Elementary Locomotion. Fig. 2 shows the vertical accelerations collected on a Nexus 5 phone when a user is performing one of the 6 elementary locomotions. When the foot strikes the ground, an unmistakable vertical acceleration is generated throughout the entire body and can be sensed by the phone no matter where it is placed and/or how it is being held, making vertical acceleration the most reliable indicator of locomotion. It is immediately clear that beyond the obvious differences in the actual values, the readings exhibit distinct patterns for every locomotion and with some analysis one should be able to identify unique signatures. In the interest of brevity, we describe in detail the process for identifying signatures for walking and running; the process is the same for strolling and rushing. Fig. 3 shows the time series of the sensor readings generated when a user is walking and running. Zooming in on the vertical acceleration, right after the heel strikes the ground, reveals a significant difference in the shapes of the respective time series – both present two peaks but while for walking the peaks are similar in size that is not the case for running. The reason for this difference lies in how most people walk and run with shoes. When running, most people [13], including the user in this experiment, land hard on their heels, which end up absorbing most of the energy. When walking, people still land on their heels but the landing is more soft and flat so the energy is transferred more uniformly on the foot, explaining the similar peaks. These basic locomotion mechanics do not depend on a person’s height, weight or gender. For example, approximately 75% of shod runners heel strike [13]. Therefore, the patterns observed in Fig. 3 are used by Acrux as signatures for walking and running, respectively, for all users. Experimental results in § 6.3 show that these signatures cut across user gender, height and weight.

Finally, as Fig. 2 shows, there is no signature for static and idle so when Acrux does not identify a particular signature it knows the user is either idle or static. To distinguish these two locomotions, Acrux uses the fact that an idle user generates far stronger readings than a completely static one.

![Figure 2: Vertical acceleration readings for every elementary locomotion.](image)

![Figure 3: Acceleration and gyroscope readings (most relevant dimensions shown) with phone in the right hand. Clear and distinct patterns emerge in the vertical acceleration readings for walking and running, respectively.](image)
2.126 Hz

3.1.2 Storing the Signatures. Let \( A_{i,k}(t) \) be the 5 s\(^4\) temporal realization of elementary locomotion \( i \) from sensor reading \( k \), where \( i \in \{1, m\} \) and \( k \in \{1, p\} \). For the current version, \( m = 4 \), for strolling, walking, rushing, running and \( p = 6 \); 3 readings from the accelerometer and 3 from the gyroscope.

- The signal is truncated, centered and zero-padded.
- Each signal is converted into the frequency domain, \( \mathcal{A}_{i,k} = \text{FFT} \left( A_{i,k} \right) \), and normalized: \( \mathcal{A}_{i,k}^{\text{norm}} \)
- All signals are entered on a matrix: \( X = (\mathcal{A}_{i,k})_{i\in[1,m],k\in[1,p]} \).

3.1.3 Real Time Elementary Locomotion Identification. When running on a phone, Acrux periodically collects sensor readings and looks for signatures so as to identify the current elementary locomotion. For this it needs to, one, decide how much data it needs for identifying a particular signature and, two, a quick and efficient way of searching the data for signatures. Fig. 3 gives an easy answer to the first question — 1 to 2 seconds of sensor readings contain more than enough patterns. The challenge is identifying them in real time.

**Matching current sensor readings to a signature:** Instead of searching the sensor readings for a particular signature, Acrux simply compares a sample of the current readings with the stored signatures (§ 3.1.2) and returns the closest match. The input readings are first processed and then converted into the frequency domain using FFT. The processing consists of truncating the signal for removing the edge effects and mean value and then zero-padding so the signal’s length is a power of two\(^5\). The readings are compared to the stored signatures using the Pearson coefficient. For two signals \( x \) and \( y \), each of length \( n \), the Pearson coefficient, \( r \), is defined as follows:

\[
 r_{\text{Pearson}}(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \cdot \sum_{i=1}^{n} y_i^2}} = \frac{\mu_x \mu_y}{\nu_x \nu_y}
\]

Once the Pearson coefficient is computed for all 6 readings, we get a scalar correlation score by multiplying with a weighting vector reflecting the relative importance of each reading in identifying locomotions. As mentioned in § 3.1.1, the vertical acceleration is the most reliable indicator of locomotion. Based on this and hours of trial and errors, Acrux implementation uses 0.4, 0.2, 0.05 for the accelerometer’s vertical, frontal and side readings and 0.2, 0.1, 0.05 for the gyroscope’s yaw, pitch and roll. If the scalar correlation score for a particular candidate locomotion is at least 0.4, it is considered a good potential fit. If no candidate locomotion scores at least 0.4, indicating no locomotion pattern in the sensor readings, it returns idle by default\(^6\).

3.1.4 Accuracy in Practice. To test the accuracy of our approach for identifying elementary locomotions, we carry the following experiment. A single user travels a 200 m distance while holding a Nexus 5 phone running Acrux and is asked to switch between the 6 elementary locomotions, as the user perceives them, every 10 s. Acrux tries to identify the elementary locomotions using readings from a fixed time window. This time window is varied from 0.25 to 2.5 s and a separate experiment is carried out for every value. We collect the data and compare offline what Acrux identified as elementary locomotion with what the user was actually doing. The data showed (graph not shown for lack of space) that a 1 s sample was enough for Acrux to identify the elementary locomotion with over 90% accuracy. When the sampling window is stretched out to 2.5 s the accuracy approaches 100%. For the performance evaluation (§ 6) we used a 1 s window as it presents a good tradeoff between accuracy and speed.

\(^5\)As is evident from Figures 2 and 3, a few seconds contain more than enough patterns. We tested and got good results with as little as 2 s but as it costs next to nothing we chose to be conservative and sampled for 5 s.

\(^6\)When the readings are close to 0 Acrux decides the current locomotion is static and does not trigger Algorithm 1.
for addressing this two-fold challenge [24, 27]. Fortunately, Acrux devices is itself erroneous. Several solutions have been proposed.

4 ESTIMATING DIRECTION

4.1 Locomotion Based Filtering

The gyroscope measurements are impacted not only by changes in direction but also by how a user moves. To filter out the impact of these movements, Acrux makes use of its locomotion signatures. When it needs to estimate direction, it first identifies the current elementary locomotion and then subtracts the yaw velocity of the particular locomotion from the current gyroscope values. Specifically, it first converts the current yaw velocity into the frequency domain and then subtracts from its highest peak the highest peak of the stored signature (already in the frequency domain § 3.1.2). This gives Acrux a filtered version of the yaw velocity that is a better indication of the actual direction changes.

4.2 Bias Correction

Let \( \hat{\theta}_m(t_n) \) denote the measured and filtered (§ 4.1) yaw velocity at time \( t_n \). Because of the bias and noise introduced by the MEMS gyroscope, we get:

\[
\hat{\theta}_m(t_n) = \hat{\theta}(t_n) + b(t_n) + \eta(t_n)
\]

where \( \hat{\theta}(t_n) \) represents the correct yaw velocity, \( b(t_n) \) represents the bias introduced by the gyroscope and \( \eta(t_n) \) represents the white Gaussian noise.

Thus, estimating the correct yaw velocity using Eq. (1) reduces to the problem of estimating the bias. We can re-write Eq. (1) in recursive form for \( i = 1, 2, 3 \ldots n \) as follows:

\[
\hat{\theta}_m(t_i) - \hat{\theta}(t_{i-1}) = \hat{\theta}(t_i) - \hat{\theta}(t_{i-1}) + b(t_i) + \eta(t_i)
\]

Taking expectations, we get:

\[
\mathbb{E} \left[ \hat{\theta}_m(t_i) - \hat{\theta}(t_{i-1}) \right] = \mathbb{E} \left[ \hat{\theta}(t_i) - \hat{\theta}(t_{i-1}) \right] + \mathbb{E} \left[ b(t_i) \right]
\]

Assuming no abrupt changes in the direction of locomotion, that is, \( \mathbb{E} \left[ \hat{\theta}(t_i) - \hat{\theta}(t_{i-1}) \right] = 0 \), leads to:

\[
\mathbb{E} \left[ b(t_i) \right] = \mathbb{E} \left[ \hat{\theta}_m(t_i) - \hat{\theta}(t_{i-1}) \right]
\]

(2)

Finally, using the fact that at time \( t_n \), \( \mathbb{E} \left[ b(t_i) \right] = \frac{1}{n} \sum_{i=1}^{n} b(t_i) \), Acrux can estimate the bias of the MEMS gyroscope as follows:

\[
\begin{align*}
b(t_n) &= n \times \mathbb{E} \left[ b(t_i) \right] - \sum_{i=1}^{n-1} b(t_i) \\
&= n \times \mathbb{E} \left[ \hat{\theta}_m(t_i) - \hat{\theta}(t_{i-1}) \right] - \sum_{i=1}^{n-1} b(t_i)
\end{align*}
\]

(3)

As base case for Eq. (3), we assume \( \hat{\theta}(t_0) = 0 \).
5 LOCALIZATION IN 3D: STAIRS, ESCALATORS, ELEVATORS

To continue tracking a user’s trajectory as she moves between floors in a multilevel building, Acrux needs to address the following three challenges. One, estimating the altitude change. Two, detecting the vertical transportation mode – stairs, escalator, elevator. Three, estimating the displacement given the user’s vertical transportation mode. In the following, we explain how Acrux addresses each of these challenges.

5.1 Estimating Altitude Changes Using the Barometer Sensor Readings

Acrux calculates the changes in altitude by leveraging the barometer sensor and adapting the barometric formula as follows:

\[ \Delta z = -\frac{\Delta p}{\rho g} \]  

where \( \Delta z \) is the differential altitude, \( \Delta p \) is the pressure differential, \( \rho \) is the air density and \( g \) is the earth’s gravity.

Unfortunately, the air density, \( \rho \), can be highly variable due to, among other things, changes in room temperature. Fig. 6 shows altitude estimations when using Equation 4 with the barometer readings from 4 different mobile devices left stationary for approximately an hour on a table in our building. The data shows that over time the barometer readings for the same exact location drifts by amounts corresponding to several meters, the equivalent of one or more floors. Clearly, one cannot rely on the absolute barometer readings for identifying the exact floor a user is on. A similar conclusion is also reached in [22, 29] which, as a result, introduce training and output from outside in the form of fingerprinting or crowdsourcing to achieve accurate floor placement.

Fortunately, Acrux’s goal is not exact floor placement but dead reckoning. It only needs to know if a user has changed floors not the exact floor she is on, which would require the exact number of floors she has changed since entering the building. A floor change using any of the vertical transportation modes takes place in a matter of seconds, time during which, as Fig. 7 shows, the change in barometer readings due to changing floors is much higher than the drift. Therefore, by analyzing altitude changes taking place over a few seconds, Acrux can tell if a user is changing floors, the vertical transportation mode § 5.2 used and the displacement between two floors § 5.3 so as to compute the next fix.

5.2 Detecting Vertical Transportation Mode

To detect the vertical transportation mode, Acrux uses the decision tree shown in Fig. 8. When it detects a high altitude change over few seconds (>1m in 2 seconds), it knows the users has taken a vertical transportation mode. It then identifies the elementary locomotion and if it is static or idle it decides the user is either on an elevator or standing on an escalator. To distinguish these two, it looks at the vertical acceleration; if it is high (> 1m/sec^2) the user is in an elevator otherwise she is standing on an escalator. If the elementary locomotion is not static or idle the user is either climbing the stairs or is walking on an escalator. To distinguish these two, Acrux uses the fact that a user who is walking on an escalator will normally change altitude significantly faster than someone taking the stairs.
5.3 Estimating Displacement for Every Vertical Transportation Mode

To generate fixes while on an elevator, Acrux only needs to update the vertical axis value. For this it simply computes the attitude change since the last fix using Equation 4. In the case of escalators, generating fixes using the attitude change only is not possible as the movement is both across the vertical as well as the horizontal axis. However, with a little research we found that the angle of inclination of an escalator is regulated and is typically 30° [3]. Using this information and simple trigonometry we can compute the displacement in the horizontal axis and continue generating accurate fixes in 3D. Finally, generating fixes while a user is taking the stairs is a bit more complex – a user taking a spiral staircase, for example, is continually changing direction while climbing. To address this, Acrux makes use of its direction estimation module in addition to the altitude change and the fact that the angle of inclination of stairs is also regulated and is typically 30°.

6 SYSTEM EVALUATION

6.1 Implementation

We implemented Acrux following a write once, run anywhere approach. Towards this, we used PhoneGap [4], a mobile development framework that enables application development for mobile devices using JavaScript. It allowed us to implement a responsive user interface in HTML5 and CSS3 and a unified back-end in JavaScript. What is more, with PhoneGap we were able to implement part of Acrux as a native process which allowed for a robust interface with the sensors. Finally, we used Couchbase [2] for building a fast and efficient database on a mobile device following the NoSQL approach.

6.2 Methodology

We evaluate Acrux in real-life experiments using 10 users, 4 different mobile devices and 4 different buildings. The mobile devices include an HTC Google Nexus 5 with Android 5.0 Lollipop, an HTC Google Nexus 9 tablet with Android 5.0 Lollipop, a Samsung Galaxy S5 with Android 4.4.2 Jelly Bean and a Sony Xperia E3 with Android 4.4 Jelly Bean. The 4 different buildings include a two-story office building, a five-story university building and two subway stations. Collecting the ground truth proved challenging as we do not have access to detailed architecture drawings of the buildings in which we carried out the experiments, the subway stations in particular. Instead we do the following. For every experiment we use two volunteers per mobile device: one carrier and one “shadower”. During an experiment, the phone carrier selects arbitrarily an end point and “walks”8 there at normal pace; along the way the “shadower” marks the floor once a second. Acrux also outputs fixes once a second. After the carrier reaches the end point, we use a meter and a protractor to measure the coordinates of every mark on the floor – the set of all these coordinates constitutes the ground truth. The elementary locomotion signatures Acrux uses throughout the experiments are from a single user, as described in § 6.3.

8When not clear from the context, we use the quotation marks to distinguish walking as in someone getting from point A to point B on foot from walking, the elementary locomotion.

6.3 Accuracy Across Different Users

A basic premise of this work is that the elementary locomotions generate unique signatures in the inertial sensor readings that do not depend on someone’s height, weight or gender. Hence we start the performance evaluation by testing this premise.

Method: Using the approach described in § 3.1.1 and § 3.1.2, we build elementary signatures for each of the 10 volunteers participating in the experiments. We select at random the signatures from one volunteer, make them Acrux’s only signatures for the rest of this experimental study and throw away the rest. Each volunteer is then asked to travel a 200 m distance while holding a Nexus 5 phone freely at hand and switching between the 6 elementary locomotions, as they perceive them, every 10 s. Acrux tries to identify the elementary locomotions using readings from a 1 s fixed time window. We collect the data and compare offline what Acrux identified as elementary locomotion with what the user was actually doing.

Results: Fig. 9 shows the correct elementary locomotion detection rates for the 10 volunteers along with information as to their gender, height and weight. We observe that, using a single signature per elementary locomotion, Acrux is capable of correctly detecting the elementary locomotions of a diverse set of users with an accuracy between 88.3% and 93.5%.

For a look under the covers, Fig. 10 shows a sampling of the vertical acceleration readings collected for 3 of the volunteers during the above experiment (including more readings from all the volunteers would make the plot illegible but similar behavior was observed). We observe remarkably similar signatures, despite the differences in gender, height and sex, which is the underlying reason for the results of Fig. 9.

6.4 Acrux vs. Traditional Dead-Reckoning

In this experiment, we evaluate Acrux’s capability to accurately measure distance, the most challenging part in dead-reckoning, and compare it against step-counting, the approach traditionally adopted for pedestrian dead-reckoning.

Comparison: Ideally, we would compare Acrux to full-fledged implementations of state-of-the-art, step-based systems [18, 27].
Unfortunately, none of these systems are available as open-source and we find it infeasible to faithfully implement them based simply on the paper descriptions. However, we can bound their accuracy using an approach we refer to as the Measured step size and compare against this bound.

[18, 27] start with a generic model assuming a linear relationship between step size and step frequency for any user. Then, the slope and constant term of the linear function are estimated on a per user basis. [18] uses a particular filtering based approach requiring the user to physically turn several times in order to converge as well as input from the user. [27] selects and improves the parameters of the linear equation using a feedback loop involving map (e.g. floor plan) matching. The sophistication of these approaches and the help from outside they need highlight the difficulty facing any step-based approach. Most important, a system assuming a linear relation between step size and step frequency and needing to learn crucial parameters as the user walks cannot be more accurate than the Measured step size approach.

**Method:** Ten volunteers are asked to “walk” as they normally would for 25 m and every step they take is measured. The average over all sizes measured for a particular user is selected as her step size. Then a volunteer is selected arbitrarily among the 10 and is asked to “walk” the same way down a boulevard for 5 km while trying to avoid turns and changes in direction – to avoid other components designed for estimating direction from affecting the outcome – using the same shoes and outfit as during the 25 m walk. Using Android’s special purpose App, we track the step count at regular intervals and store it along with a timestamp for offline processing. The error for step-counting is computed by comparing the ground truth distance to the product of the step count with either the step size of the volunteer taking the 5 km walk – the Measured step size approach – or the average step size over the 10 volunteers – the Group step size approach.

**Results:** Fig. 11 (top graph) shows that after 5 km, Acrux’s deviation from the truth is 8.33 m while step counting using the Measured step size approach deviates by 52.29 m – over 6.5 times as much as Acrux. This is due to the fact that no two steps are the same even if everything else – user, outfit, environment – is kept the same. Thus, the Measured step size approach will suffer from errors, which considering the large number of steps one takes, will accumulate fast over distance. Step counting using as step size the average over 10 people gives the unacceptable result of 414.8 m deviation from the true distance.

For a better understanding, we zoom in on the data for which all approaches exhibit an error of just over 1 m. As the bottom graph in Figure 11 shows, Acrux can dead-reckon for 557 m before the deviation from the truth reaches 1 m, while the Measured and Group step size approaches hit the same error after 19 and 99 m, respectively.

### 6.5 Large Scale Measurements in the Wild

In this experiment, we evaluate Acrux’s localization accuracy across a large and diverse set of paths, users and indoor locations.

**Method:** We select arbitrarily 46 different start-end pairs in 4 locations – 32 at an office building, 12 at a university building
54. Figure 13: Acrux “trying to keep pace” with a user following a complex trajectory inside an office building. The whole trajectory is around 400 m.

and one at each subway station (carrying out experiments at a busy subway station proved challenging). The 10 phone carriers participating in the experiment are each given the same phone9, a Nexus 5, and are asked to go from the start to the end point “walking” as they would in a normal situation. Similarly, they are told to hold the phones as they normally would – in hand or pocket. Once the experiment concluded we had data from 460 trajectories of lengths between 80 – 120 m.

Results: Fig. 12 shows the cumulative distribution function (CDF) of the error for Acrux across all 460 trajectories. The data shows that Acrux attains a median and 98% percentile error of 1.2 and 3 m, respectively.

6.6 Catch Me If You Can

In this experiment, we evaluate Acrux’s ability to map a complex user trajectory inside an office building.

Method: A single user is invited to follow a purposefully complex trajectory (Fig. 13(b)) inside an office building while carrying a smartphone in her pocket. In particular, the user is invited to take many turns, a major source of localization errors, and to switch pace between strolling, walking and even running. At some point she pauses to talk to a colleague. The whole trajectory is about 400 m.

Results: Fig. 13(a) shows that despite the challenging scenario, Acrux is able to successfully map the true trajectory. The median error is only 0.7 m and never exceeds 3 m.

6.7 Going Upstairs

In this experiment, we evaluate Acrux’s ability to localize a user as she moves across different floors inside a building.

Method: A user selects one of the 4 devices available and takes the stairs, shown in Fig. 14, to the second floor of the office building. In this particular experiment we did not compute a ground truth and instead relied on visual verification.

Results: Fig. 14 shows the trajectory mapped by Acrux superimposed on the picture of the staircase. Considering the small size of the stair riser, 15 cm, one can visually verify the high accuracy with which Acrux follows a user’s every step as she is climbing up the stairs.

6.8 Impact on Battery Life

Finally, we evaluate Acrux’s performance in terms of energy consumption, a crucial metric for an application running on battery powered devices.

Method: Starting with fully charged batteries, Acrux is left running and generating fixes 1/sec on a tablet and smartphone until the batteries die. Meanwhile, we measure its power consumption using the Android App Power Tutor [1]. To contextualize Acrux’s energy performance, we compare it to BaselineRF, a simple application we developed to establish a baseline for the energy consumption of RF-based localization schemes. Every time Acrux generates a localization fix, BaselineRF simply performs a passive (listen only) WiFi scan. An actual RF-based localization scheme needing to generate a localization fix would most probably have to perform more tasks than simply looking for access points, thereby consuming more energy than BaselineRF.

Results: Table 1 shows that Acrux draws less power than BaselineRF. This excellent energy performance is due to the fact that Acrux does not make use of the WiFi card, relying instead on sensors, such as the accelerometer, shown to be energy efficient [15].

7 RELATED WORK

While the literature on indoor localization is rich most works fall into four overlapping categories.

RF-based solutions: A majority of indoor localization solutions rely on the WiFi transceivers on mobile devices and ubiquitous availability of WiFi hot spots [5, 14] to generate accurate fixes indoors. Based on how the distance to the access points is estimated,
We presented Acrux, an indoor localization system that can dead-reckon locomotions based on things such as walking surface, crowd density, localization with the minimum outside input – a single fingerprint. Inspired by how humans perceive pedestrian locomotion indoors an innovative algorithm for estimating the speed of locomotion trains outside of GPS coverage. This is accomplished thanks to important, long enough to cover most distances traveled by pedestrians several times improvement over the state-of-the-art, and more robustness while the system’s performance relies on complex computations required for generating every localization fix for every device in the range of a particular WiFi network.

RF-based solutions with dedicated infrastructure and/or hardware: Using dedicated hardware and/or infrastructure can lead to indoor localization systems with cm-level accuracy [28], however, as with the fingerprinting solutions it creates a high barrier to entry that so far no solution has been able to overcome.

Non RF-based solutions: Given the increasing sophistication of the smartphone hardware, several solutions have proposed taking advantage of the inertial sensors to build dead-reckoning based localization schemes [6, 18]. However, all the dead-reckoning based schemes require periodic recalibration [18, 27].

Crowdsourcing based solutions: Most solutions belonging in the three previous categories could be improved by using crowd-sourcing [8, 9, 23]. While these solutions are very promising they are hindered by a chicken-egg problem – early adaptation relies on the system’s performance while the system’s performance relies on large scale adaptability.

8 CONCLUSION AND FUTURE WORK
We presented Acrux, an indoor localization system that can dead-reckon with meter accuracy for several hundred meters, a several times improvement over the state-of-the-art, and more important, long enough to cover most distances traveled by pedestrians outside of GPS coverage. This is accomplished thanks to an innovative algorithm for estimating the speed of locomotion inspired by how humans perceive pedestrian locomotion indoors combined with careful measurement based analysis. We have developed Acrux on the Android OS and through experiments in 4 different buildings shown that it can offer meter level indoor localization with the minimum outside input – a single fix.

This work is not the last chapter on Acrux. While this version focused on the 6 basic elementary locomotions, future work will focus on breaking down the elementary locomotions into sub-locomotions based on things such as walking surface, crowd density, time of the day, geography, etc. Coupled with more sophisticated classification algorithms, this has the potential to significantly improve the robustness and reliability of Acrux.

REFERENCES