You Are Sensing, but Are You Biased? A User Unaided Sensor Calibration Approach for Mobile Sensing

ANDREAS GRAMMENOS, University of Cambridge and The Alan Turing Institute
CECILIA MASCOLO, University of Cambridge and The Alan Turing Institute
JON CROWCROFT, University of Cambridge and The Alan Turing Institute

Mobile devices are becoming pervasive to our daily lives: they follow us everywhere and we use them for much more than just communication. These devices are also equipped with a myriad of different sensors that have the potential to allow the tracking of human activities, user patterns, location, direction and much more. Following this direction, many movements including sports, quantified self, and mobile health ones are starting to heavily rely on this technology, making it pivotal that the sensors offer high accuracy.

However, heterogeneity in hardware manufacturing, slight substrate differences, electronic interference as well as external disturbances are just few of the reasons that limit sensor output *accuracy* which in turn *hinders* sensor usage in applications which need very high granularity and precision, such as quantified-self applications. Although, calibration of sensors is a widely studied topic in literature to the best of our knowledge no publicly available research exists that specifically tackles the calibration of mobile phones and existing methods that can be adapted for use in mobile devices not only *require* user interaction but they are also *not* adaptive to changes. Additionally, alternative approaches for performing more granular and accurate sensing exploit body-wide sensor networks using mobile phones and additional sensors; as one can imagine these techniques can be bulky, tedious, and not particularly user friendly. Moreover, existing techniques for performing data corrections post-acquisition can produce inconsistent results as they miss important context information provided from the device itself; which when used, has been shown to produce better results without a imposing a significant power-penalty.

In this paper we introduce a novel multiposition calibration scheme that is specifically targeted at mobile devices. Our scheme exploits machine learning techniques to perform an adaptive, power-efficient auto-calibration procedure with which achieves high output sensor accuracy when compared to state of the art techniques without requiring any user interaction or special equipment beyond device itself. Moreover, the energy costs associated with our approach are lower than the alternatives (such as Kalman filter based solutions) and the overall power penalty is $< 5\%$ when compared against power usage that is exhibited when using uncalibrated traces; thus, enabling our technique to be used efficiently on a wide variety of devices. Finally, our evaluation illustrates that calibrated signals offer a tangible benefit in classification accuracy, ranging from 3 to 10%, over uncalibrated ones when using state of the art classifiers; on the other hand when using simpler `SVM` classifiers the classification improvement is boosted ranging from 8% to 12% making lower performing classifiers much more reliable. Additionally, we show that for similar activities which are hard to distinguish otherwise, we reach an accuracy of $> 95\%$ when using neural network classifiers and $> 88\%$ when using `SVM` classifiers where uncalibrated data classification only reaches $\sim 85\%$ and $\sim 80\%$ respectively. This can be a make or break factor in the use of accelerometer and gyroscope data in applications requiring high accuracy e.g. sports, health, games and others.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile devices**; **Mobile devices**; *Smartphones*;

Authors' addresses: Andreas Grammenos, University of Cambridge and The Alan Turing Institute, Cambridge, UK, CB3 0FD, ag926@cl.cam.ac.uk; Cecilia Mascolo, University of Cambridge and The Alan Turing Institute, Cambridge, UK, CB3 0FD, cm542@cl.cam.ac.uk; Jon Crowcroft, University of Cambridge and The Alan Turing Institute, Cambridge, CB3 0FD, UK, jac22@cl.cam.ac.uk.

2474-9567/2018/3-ART11 $$15.00
https://doi.org/10.1145/3191743

Additional Key Words and Phrases: Sensor, Calibration, IMU, Mobile Devices

ACM Reference Format:
Andreas Grammenos, Cecilia Mascolo, and Jon Crowcroft. 2018. You Are Sensing, but Are You Biased? A User Unaided Sensor Calibration Approach for Mobile Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 11 (March 2018), 26 pages. https://doi.org/10.1145/3191743

1 INTRODUCTION

Mobile devices are almost ubiquitous today, as they serve as invaluable convenience devices accompanying us almost everywhere. Applications for such devices are numerous, from digital wallets to social tools. In addition, these devices come equipped with a myriad of sensors that can be used to perform navigation, activity recognition, workout tracking, just to name a few. Quantified self movements as well as mobile health applications have increasingly started to rely on these mobile sensors to augment the data they collect.

However, the quality of the sensors in such devices is not usually comparable to professional grade ones mostly due to power and economic costs that are associated with higher quality sensors. This results in sensing data which is often biased and imprecise due to the manufacturing imperfections, temperature differences, electronic noise, external interference, amongst other things.

It is therefore imperative tofi nd schemes and procedures able to keep the readings from these sensors as clean and as accurate as possible while respecting a very tight power budget. Kalman-filtering based methods have been employed successfully for bias removal in sensors [27, 42] but require *constant* processing and thus their energy cost is significant, thus making them not suitable to use in our setting [31]. Previously presented non-Kalman based schemes [21, 30] require manual parameter tweaking and/or special equipment and they all involve at least some user interaction and are not adaptive (i.e. one-shot calibration). Concretely, such techniques in mobile devices makes the experience tedious and user adherence to the calibration procedure is not always guaranteed. Furthermore, it has to be noted that although calibration has been well studied in many domains as is evident above, no publicly available work has specifically targeted mobile devices calibration nor attempted to quantify the power implications its use might have.

In this paper we present a scheme able to perform sensor calibration automatically and in an energy efficient manner without requiring any user interaction. We constrain our presentation here to two sensors specifically, accelerometer and gyroscope, however some aspects of the approach can be generalized to other types of sensors. The approach takes advantage of the fact that when a device is stationary the only force applied to it is gravity and that devices are indeed stationary in a number of daily phone activities such as charging and data syncing. In these positions the accelerometer and the gyroscope are only affected by gravity. This force is *constant* and, while its value might have disparities across devices, it can be used to *calibrate* the sensors successfully. Our calibration method is completely *transparent* to the user and the application. Additionally, it is able to tune both sensors using the same sensor model but gives thefl exibility to independently compute the model parameters for each while also maintaining its calibration accuracy over time. This is achieved by *periodically* testing the quality of the calibration thus ensuring it remains acceptable while making corrections otherwise. For calibrating the accelerometer we adapt the method outlined in [13] to a user unaided version, which ensures quadratic convergence, requires a few iterations, and is based on the same optimization principles (using gravity), while being computationally efficient. For the gyroscope, we use a variant of the method presented in [12] which employs sensor fusion using as a reference a calibrated accelerometer in order to estimate the rotation between two *stationary* positions. In addition to the previously discussed schemes, we also present an improvement to the variants mentioned above that uses sensor fusion exploiting location services to perform a better estimate of g, as it varies by geographical position and altitude. In most methods in literature this is normally set to a default value of g (9.80665) [15]. We use a location assisted technique for estimating a more representative g value, as

described in Section 4.5; this in turn, helps us to reduce the bias error further. Finally, suitable activities that can be used for calibration or quality assessment are ones that involve the mobile device being *stationary*. We detect them dynamically on device using a combination of gyroscope and accelerometer measurements over a period of time. The contributions of this paper can be summarized in the following points:

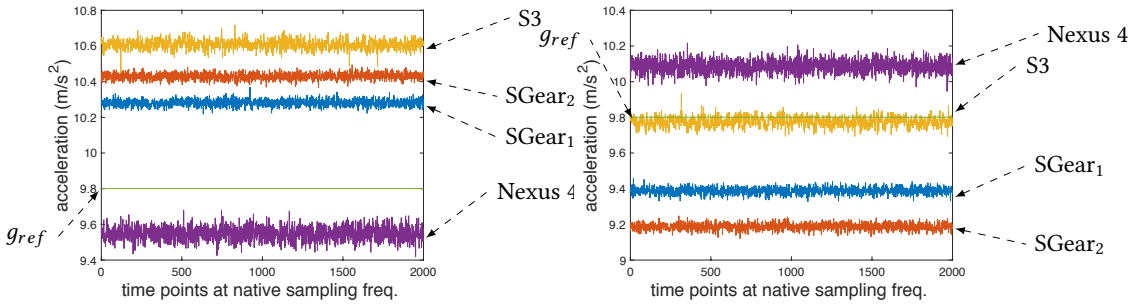
- An adaptation of existing sensor calibration methods resulting in the introduction of a multiposition-based calibration scheme specifically targeting mobile devices which are extended to perform the calibration *completely user unaided*, while improving on their performance by using a better g-value estimation.
- A *dynamic stationary detection component* able to decide which states to be used for calibration and when it should be performed.
- An evaluation using activity datasets released alongside paper [36] which shows that using calibrated traces provides a tangible classification benefit, ranging from $3-12\%$, especially in cases where activity traces are similar to each other. Using calibration resulted in having, in most cases, a near perfect classification accuracy of $> 95\%$ whereas the uncalibrated traces, in highly similar activity traces, had a classification accuracy of $\sim 85\%$ when using neural networks and $\sim 80\%$ when using SVMs.
- An energy consumption evaluation on a real device showing be to using *at least* 2.5 times *less* power when compared with a Kalmanfi lter based approach.
- Overall, power-wise our scheme has $< 5\%$ overhead during real-world usage while providing tangible classification benefits when comparing against the use of uncalibrated traces.

The ability to improve the classification in a power efficient way, especially for very similar classes of activities, has the potential to unlock the level of precision and granularity needed by continuous sensing applications (e.g. sports, health, quantified-self) to be trusted more widely.

2 MOTIVATION

In mobile sensing one of the key components that is widely employed for performing activity detection is the Inertial Measurement Unit (from now on IMU); in addition, to sensing many application use the IMU for attitude estimation such as Mobile games, AR applications, and others. Recently though, there has been an increasing number of specialized applications (e.g. sports, mobile health, quantified-self) that are starting to rely more and more on continuous mobile phone sensing for a number of use-cases ranging from personalized coaching assistance to medical tracking. In all of these cases it is assumed that the quality of the inferences derived from the data is high. However, the accuracy of low-cost sensors that mobile devices have can be quite limited and often hinders the quality of the user activity classification [36]. Additionally, although calibration is well-studied in different contexts for specialized applications (e.g. military, ship/airplane navigation), to our knowledge, it has not been studied specifically in the context of mobile devices as a whole but only in specific chips [13, 34], or other domains [24, 42]. On the other hand, there have been a number of calibration techniques that have been devised as part of larger projects targeted at mobile devices (e.g. [21]) but the focus was not given to the calibration itself, how it performed, any potential power implications, or how it improved the quality of the data when using uncalibrated traces. Moreover, one could adapt calibration techniques used in other domains such as ones cited before but these have their own limitations and their overall performance has not been evaluated in the mobile device setting as there can be a number of systems challenges involved. For example in the context of mobile devices the overall cost of calibration must be evaluated and that is ignored in all of the calibration related papers; additionally, as said previously none of the method described are adaptive which presents interesting challenges on how to performing these tasks without impacting the overall battery consumption in a detrimental way.

Concretely, and in order to demonstrate this problem more clearly we give a few examples of how this can manifest. The simplest one shows the variation of the value of the accelerometer sensed data when devices are



(a) static, phones facing front (b) static, phones facing bottom

Fig. 1. Sensor Bias examples with stationary phone: a) phones lying on back. b) phones lying on front.

standing still (in m/s^s). Figure 1a and Figure 1b show the modulus [1] value of the accelerometer readings of four different mobile devices from [36] while static, with the phone lying on a surface on its back and on its front, respectively. From the following graphs it is evident that there is a significant deviation in the observed values amongst the different devices when compared against the expected ones, which should in both cases be $1g$ or ~9.81. Furthermore, in Figure 1 we can see that two *distinct* types of biases are at play and these contribute to the total error. One stems from the *actual* sensor biasing error (deterministic error) and is mainly displayed as the shift from g, while the other is the electronic noise (stochastic error) which affects the sensor. Finally, another noteworthy observation is that the bias changes if the phone changes orientation.

Unfortunately, this bias which is introduced due to the heterogeneous nature of the data-sources makes it increasingly difficult to classify user activities appropriately [23, 36]. This is better shown in Figure 2 as we have two accelerometer traces taken using the same device and the same subject: thefi rst one is for regular walking whereas the second is for climbing up stairs; these arguably look significantly similar.

Attempting to perform this "correction" or "pre-processing" after data collection is *very* difficult and can lead to inconsistent results. It is important to note that in such cases inference classification alone will not always be able to deal with these errors and variations.

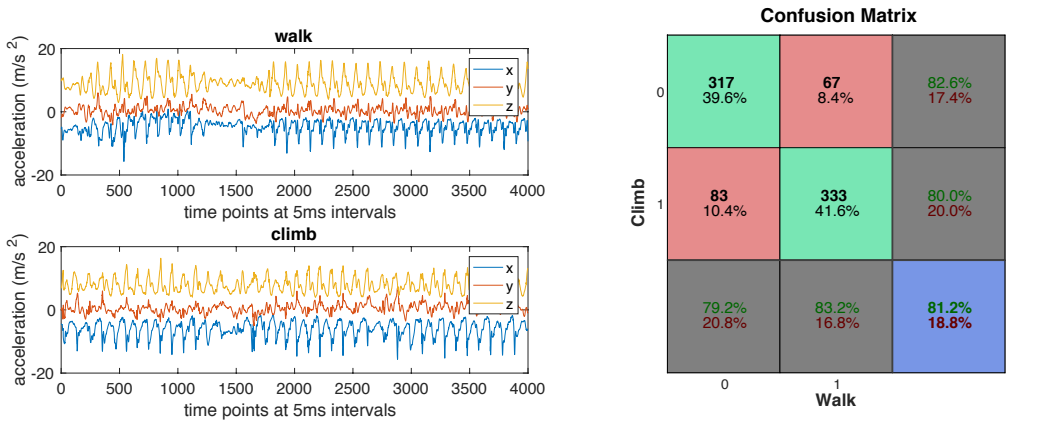
3 RELATED WORK

Calibration of Inertial Measurement Units has been an extensively studied topic for many years. Commonly, an IMU includes two major components, an accelerometer and a gyroscope; recently though, with technology advances, most commercially available IMU's started including a magnetometer component as well. An accelerometer is responsible for determining the linear acceleration, a gyroscope is responsible for measuring angular velocity, and in turn orientation while,fi nally, the magnetometer is responsible for measuring the strength of the surrounding magneticfi eld. In this section, we will initially describe general principles of calibration techniques, then we will emphasize on recent auto-calibration techniques, andfi nally, we will conclude outlining and comparing previously presented calibration schemes in mobile devices.

3.1 IMU Errors

Unfortunately, all IMU sensors suffer from various types of errors which can have detrimental effect its accuracy and can be roughly be divided into two distinct categories: deterministic and stochastic. Deterministic errors are

[1] The modulus can be viewed as the $\mathbf{L}_2$ norm of the accelerometer values



(a) Accel. traces of walking (top) vs stair climbing (bottom) (b) Confusion matrix when using an SVM classifier

Fig. 2. Similar traces of walking & stair climbing

the ones that can be detected in their entirety by monitoring IMU output and can be attributed to manufacturing processes, component variation, and/or material differences. On the other hand, stochastic errors cannot be exactly measured based on monitoring the system's output and can only be approximated statistically; such errors usually are the electronic noise interfering with the system's output and is assumed to be Gaussian in nature [2]. Although both types of errors contribute to the IMU output signal quality degradation, deterministic errors are the ones that contribute significantly more in practice [13, 30] and hence, are those which calibration techniques try to address. Deterministic errors, which are the errors calibration tries to address, can be of three types: Bias, Scaling factor, and Non-orthogonality Misalignment errors which are elaborated and described below.

3.1.1 Bias Errors. Bias can manifest as the deviation from a known reference value when no external stimuli or force is applied to the sensor. Due to this, the ideal conditions of evaluating and isolating this type of deterministic error are stationary positions. Additionally, it has to be noted that temperature can also play a role on how bias offset behaves over large temperature differences; this has been explored in previous works [2] and appears to be deterministic in nature.

3.1.2 Scaling Factor Errors. Scaling errors can be identified as the ratio of change in the output to the change of the input and are generated evaluated as the slope of the bestfi t straight line which can sufficiently describe the sensor input-output data; this difference between reference and actual scaling is termed as the scaling factor errors [2].

3.1.3 Misalignment Errors. Each component included in the IMU has its own reference origin point and should be the same for each of the components. Additionally, for each of the components its reference axes must be orthogonal to each other. Unfortunately, due to mounting point differences, manufacturing and/or component variation misalignment errors are introduced; this means that the the axes are not orthogonal to each other and their reference points are not aligned; these again are deterministic errors and can be amended by performing calibration.

3.2 Calibration Methods

Calibration methods can be divided into two groups where the distinction is made if there is a requirement for using any additional equipment (apart from the IMU itself) or not. Moreover the evaluation of the various calibration schemes can be viewed as the trade-off between complexity, accuracy, and ease of execution which can be optimized based each particular use-case or setting. Traditionally, the IMU calibration procedures are performed using high-end measurement equipment in order to obtain the calibration parameters that are then hard-coded into the IMU itself. This process involves mounting the IMU into an attitude controlled turntable with each of the IMU axis perfectly aligned with those of the instrument. Then a series of scripted trials is performed in order to assess and estimate the aforementioned calibration parameters accurately [38]. Although these methods can achieve great results, they have a number of significant drawbacks making their use much more difficult. Firstly a major drawback is cost and this is due to having the requirement of using specialized, expensive equipment in order to perform and assess the calibration. Additionally, these methods are labour intensive which can only be performed in a controlled environment; hence barring their use for in-field calibration, which is another important requirement of modern applications (e.g. Drones) [24].

Naturally, more and more applications started to have the need of reliable, cheap, and hassle-free in-field calibration techniques (e.g. mobile phones, activity trackers, wearables and so on) hence, this sparked a research interest which led to the inception of such methods. Thefi rst in-field calibration procedure was introduced in [11] and required the user to place the IMU in six-different positions in order to be calibrated. The principal idea of such schemes is to use some known reference values which manifest in certain positions in order to be able to estimate the calibration parameters in-field, without the need of any external equipment. It has to be noted that although no external equipment is required, all of the currently known calibration schemes require at least some degree of user interaction and are not transparent to the user. Finally, both calibration are *one-shot* procedures and are not able to perform an automatic re-calibration when there is a need to do so.

3.3 Recent IMU Auto-Calibration Techniques

A number of novel methods have been introduced that require no additional equipment to perform the calibration and are based on the observation that when an object is stationary the *only* force applied to it is gravity. By using the value of gravity acceleration, g, optimization methods have been devised which create reliable calibration schemes avoiding external equipment. Some methods focus solely on the accelerometer calibration [13, 41] whereas others are able to calibrate both the accelerometer and the gyroscope [24, 42]. For example [13] uses k random stationary states in order to perform the calibration without requiring any external equipment. Methods like [24] and [34] attempt tofi rst calibrate the accelerometer and then use it as a reference point for the gyroscope. Finally [42] explores the inherent relation between the dot product of sensor and earth physical signals but this method fails to take in account and compute the misalignment matrix.

Notably, all of the aforementioned works require at least some degree of user interaction; for example [13, 24] requires the user to place the device in a number of stationary locations. Additionally, none of these methods can automatically re-calibrate the IMU as needed nor take in account the *variability* of gravity force over different locations and altitude. Finally, most of these methods do not quantify the overall power cost of the calibration procedure, which in our context is a very important consideration.

3.4 Calibration of the IMU in Mobile Phones

To our knowledge there have been no prior publicly papers authored to tackle calibration of IMU's *specifically* in mobile phones. Although, there are papers that use calibration as a way to improve the quality of their data, they suffer from the same issues defined above, namely that they require at least some user interaction or are one-shot calibration schemes as seen in [21]. Of course, calibration has to be performed again over-time since the drifts in

time, temperature, and sensor wear all affect the IMU readings over time in a negative way [2, 34]. *The novelty of this work is that it attempts to alleviate those limitations and provide a calibration procedure specifically targeted at mobile devices that is completely user unaided while also taking in account the variability of the gravity force over different geographic locations and altitudes.*.

4 METHODOLOGY

We will now explain the details of our methodology. Wefi rst introduce our generic sensor model, used for both sensors. We then introduce the user unaided calibration procedure for the accelerometer and gyroscope. We also describe how we detect the stationary states needed for the calibration process, estimate a better g-value and, finally, how we remove the electronic noise from our output traces.

4.1 Generic Sensor Model

Our *generic* sensor model will be used to express the readings of the following *triaxial* components of the accelerometer and the gyroscope, which are part of the IMU of a device. This model can be easily extended to any sensor that requires scaling (S matrix) and absolute bias removal (B vector); it has to be noted that the model is not limited to just three dimensions. In our work we apply this correction model, as a proof of concept, to the output of the accelerometer and gyroscope with good results.

We express each one of the sensors as a three dimensional vector with components reflecting the effect in each one of the three axes separately; this can be expressed mathematically as vectors in $\mathbb{R}^3$ representing the three dimensions x, y and, z, respectively. Concretely, at any given time t the IMU output can be modeled as a 3×2 matrix, O_t. This matrix is comprised of two distinct columns, one for each IMU component (A_t, G_t) and is expressed as shown in Equation 1.

$$O_t = \begin{bmatrix} A_t & G_t \end{bmatrix} \tag{1}$$

This representation makes it easy to calibrate each sensor *independently* using the best possible scheme for each. In addition, it allows us to perform a *linear* transformation when applying the bias and noise correction to the received raw values of each component, thus making the computation more efficient. In the following sections we will describe the principal ideas used to calibrate each one of the IMU components. We will assume that each of the three components is affected by *two* distinct types of error, one which is the actual bias occurring due to various reasons (e.g. hardware imperfections, manufacturing process) to the electronic noise that affects the IMU output in the form of Additive White Gaussian Noise, known as AGWN [5, 27, 37].

4.2 Accelerometer

Our accelerometer model is based on a linear model [26] which uses a scaling matrix and a scalar offset vector to correct the output. We will consider the case where we have a *triaxial* accelerometer and its acceleration values are expressed, at any given time t, as $A_t^T = [a_{t,x}, a_{t,y}, a_{t,z}]$ which show the acceleration along each axis with respect to the device's *local* coordinate system. In addition, we will use the most common representation of axis positions which is almost ubiquitous amongst different devices and operating systems; although, it has to be noted that the exact point of origin is dependent on where the IMU sensor is located within the device. Finally, the mathematical model that is sufficient in describing the accelerometer output values is shown in Equation 2.

$$A = S(R_a - B) \tag{2}$$

Where S is the scale factor matrix in $\mathbb{R}^{3\times3}$, R_a is the raw accelerometer values and, B the bias correction vector which are both in $\mathbb{R}^3$. Let us now begin byfi rst defining what an *ideal* accelerometer is.

Definition 4.1. When the *ideal* accelerometer is *not* moving the acceleration modulus across all its axes is equal to g, where g is gravity acceleration measured in m/s^2. □

Our method is based on a variation of the method introduced [13], which we adapted and optimized for mobile devices, and user input independent, while also using a better g-value estimation (as described in Section 4.5). To perform calibration and to calculate the bias estimates we will *always* assume that in valid calibration states the mobile device is in *static* position; hence its *real* acceleration (a) is equal to zero. Based on these assumptions we derive Equation 3 where $||A||_2$ is the L_2 norm of the accelerometer values, b_ϵ is the error due to sensor bias and, n_ϵ is the error due to electronic noise. An analytical derivation is presented in Appendix A).

$$||A||_2 = g \Rightarrow ||A||_2^2 - g^2 = b_\epsilon + n_\epsilon \tag{3}$$

In order to compute the required model parameters, the sensor must be placed N times in a *static* orientation; in each of the orientations the sensor output is evaluated and the error ϵ at t-th iteration is equal to the following:

$$\epsilon_t = \sum_{i=x,y,z} \left\{ \sum_{j=x,y,z} \left[S_{ij}(R_{j,t} - B_j) \right]^2 \right\} - g^2 \tag{4}$$

The total accumulated mean squared error over all the measured orientations is equal to the summation over all measured errors values divided by the total number of measured *static* orientations N as is shown below:

$$\mathbf{E}_\alpha = \frac{\sum_{i=i}^{N} \epsilon_i^2}{N} \tag{5}$$

The number of *stationary* positions required to successfully converge is usually under 10, which aligns with thefi ndings in [13]. As noted previously there are *two* types of error, one that is associated with the *bias* and one that is associated with the effect of *electronic noise*.

4.2.1 Bias error removal. In order to remove the bias, b_ϵ from the accelerometer we *minimize* the non-linear function E_a defined in Equation 5 which, depending on configuration can have either 12 or 9 distinct parameters. This is achieved by parameterizing E_α with v and v' respectively; where v is defined as:

$$v = [B_x, B_y, B_z, S_{xx}, S_{xy}, S_{xz}, S_{yz}, S_{yy}, S_{yz}, S_{zx}, S_{zy}, S_{zz}]$$

and v' is defined as:

$$v' = [B_x, B_y, B_z, S_{xx}, S_{yy}, S_{zz}, S_{xy}, S_{xz}, S_{yz}]$$

The number of parameter depends on whether we want to impose a symmetry constraint on the scale matrix ($S = S^T$), This constraint essentially means that the cross-axis factors are the same for all three axis[26]. To minimize function E_a we employ a damped version of the Gauss-Newton non-linear optimization procedure as outlined in [22]; starting with a "good set" of initial values which get updated until convergence using Equation 6.

$$v_{t+1} = v_t - \delta \mathbf{H}^{-1}(v_t)\mathbf{J}(v_t) \tag{6}$$

where v_t are the values at the t-th iteration of the vector which contains the parameters to be optimized[2], δ is the damping factor ($\delta < 1$). Finally, $\mathbf{H}$ is the Hessian matrix and $\mathbf{J}$ is the Jacobian vector defined as shown in Equation 7.

[2]The number of which depends on the number of parameters used, either 9 or 12.

$$\mathbf{H}(v_t) = \left\{ h_{ij} = \frac{\partial^2 E}{\partial v_{t,i} \partial v_{t,j}} \right\}, \qquad \mathbf{J}(v_t) = \left[\frac{\partial E}{\partial v_{t,1}}, ..., \frac{\partial E}{\partial v_{t,9}} \right] \tag{7}$$

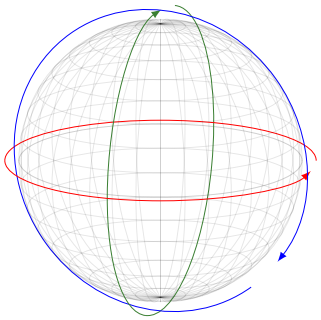
Ideally, initial "good" values, would have to be provided by the IMU manufacturer but in our setting that is obviously not possible; hence, we seed our scheme with some empirically well-performing values that in practice work well, which are discretionary. These parameters are *iteratively* updated when we detect a suitable position from which to take measurements from; this is when the mobile device is *static* as we previously mentioned. The constraint that, if satisfied, shows convergence is shown in Equation 8.

$$max \left\{ \left| \frac{v_t - v_{t-1}}{(v_t + v_{t-1})/2} \right| \right\} < \varepsilon \tag{8}$$

where ε is set equal to 10^{-6}. The calibration procedure requires few iterations to achieve convergence and the usual number of iterations being around ten (10). The total time taken for the calibration tofi nish depends on each particular users' patterns but initial convergence normally happens within one day. This was evaluated by measuring the number of potential *stationary* states (as shown in Section 4.4) within a period of 12 hours while regular device usage over a normal working day for two days.The amount of recorded states were 11 and 13 respectively, which are usually more than enough for our calibration procedure to converge successfully.

4.3 Gyroscope

Let us now shift our focus onto the gyroscope, which has *three* different axes measuring the respective rotation along each axis in radians per second (rad/s) as shown in Figure 3. In gyroscopic systems, the most *significant* source of error is the *random drift bias*. To alleviate this one could trivially measure and average the static gyroscopic signals over time; however due to the unpredictable, random, drift this can lead to major errors in computed orientation accuracy which add up over time. For our gyroscope calibration scheme we will use a variation of the method described by [12], that exploits the calibrated accelerometer as well as a better g-estimation as described in Section 4.5, which we will now describe.



Rotation labels: x, pitch, y, roll, z, yaw

Fig. 3. Gyroscope rotation axes

Practical MEMS-based IMUs use information from *absolute* orientation sensors, such as the accelerometer, in order to correct the previously mentioned drifts. Random gyroscope drifts can be modelled using the Allan

variance, σ_α^2 [9, 10], which measures the variance of the difference between consecutive interval averages and was originally used to model clock drifts. The gyroscope model used is shown in Equation 9.

$$G = S_g(R_g - B_g) \tag{9}$$

Where S_g is the scale factor matrix in $\mathbb{R}^{3\times3}$, R_g is the raw gyroscope values and, B_g the bias correction vector both in $\mathbb{R}^3$.

We can safely *disregard* the gyroscope bias, B_g, because it is averaged out during the calibration procedure since we process these values in batches. Thus, for our calibration purposes, thefi nal gyroscope model is defined as:

$$G = S_g R_g \tag{10}$$

Our calibration target is tofi nd the *nine* parameters comprising the S_g matrix that minimize our cost function, E_g, defined later on. More sophisticated methods exist that take the bias into account but these methods are not significantly better and use Kalman-based approaches which are prohibitively expensive, power wise, in our setting (see our evaluation results).

Let us define our gyroscope calibration cost function, which will be based on Definition 4.1, butfi rst we need to define an operation which converts a number of gyroscope measurements $G_{1,\ldots,n}$ using the initial gravity vector to the gyroscope gravity vector; concretely we need to define the following operation:

$$u_g = f(G_{1,\ldots,n}, u_0) \tag{11}$$

f can be any orientation integration algorithm that calculates orientation through the integration of the angular velocities $G_{1,\ldots,n}$. Quaternions are used as the method to represent orientation which we calculate using the second-order numerical integration algorithm from [16]. At each time-step the current quaternion q_t is related to the previous one, q_{t-1}; this relationship is modeled in Equation 12.

$$q_t = \left[cos\left(\frac{1}{2}|\delta\beta|\mathbf{I} + \frac{1}{|\delta\beta|}sin(\frac{1}{2}|\delta\beta|)\mathbf{C}\right)\right]q_{t-1} \tag{12}$$

where $\mathbf{I}$ is a 4×4 identity matrix while $\delta\beta$, Δt, $|\delta\beta|$ and $\mathbf{C}$ are defined as is shown below:

$$\delta\beta = G_{1,\ldots,n}, \qquad \Delta t = [\delta\beta_x \quad \delta\beta_y \quad \delta\beta_z]^T,$$

$$|\delta\beta| = \sqrt{\delta\beta_x^2 + \delta\beta_y^2 + \delta\beta_z^2}, \qquad \mathbf{C} = \begin{bmatrix} 0 & \delta\beta_x & \delta\beta_y & \delta\beta_z \\ -\delta\beta_x & 0 & \delta\beta_z & -\delta\beta_z \\ -\delta\beta_y & -\delta\beta_z & 0 & \delta\beta_x \\ -\delta\beta_z & \delta\beta_y & -\delta\beta_x & 0 \end{bmatrix}$$

From the angular velocities G_t measured by the gyroscope at every time-step, $\delta\beta$ is constructed and used in Equation 12 in order compute the current quaternion q_t. Each quaternion is comprised of four elements, $q_t = [a, b, c, d]^T$ and can be calculated using q_0 and $\delta\beta$. The rotation matrix $\mathbf{R}_q$ can obtained from each quaternion using Equation 13.

$$\mathbf{R}_q = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2(bc + ad) & (2(bd - ac) \\ 2(bc - ad) & a^2 - b^2 + c^2 - d^2 & 2(cd + ad) \\ 2(bd + ac) & 2(cd - ab) & a^2 - b^2 - c^2 + d^2 \end{bmatrix} \tag{13}$$

Now, having calculated the rotation matrix R_q and u_0 we can obtain the gyroscope gravity vector u_g using Equation 14.

$$u_\alpha = \mathbf{R}_q u_0 \tag{14}$$

Finally, the cost function E_g is defined as the summation of the squared difference of u_g and u_a as is shown in Equation 15.

$$E_g = \sum_{k=0}^{K-1} ||u_\alpha - u_g||^2 \tag{15}$$

Intuitively, we would assume that under ideal conditions $u_\alpha - u_g = 0$, but this not the case, hence we use the result to calculate the residual error at each iteration. Finally, since we are using the accelerometer measurements in order to calibrate the gyroscope this results that both of these components have the *same* reference frame.

4.4 Detecting Stationary States

Stationary state detection is a major component of our method as it enables the calibration to remain *completely* transparent to the user. Although, it has to be implemented intelligently as naive schemes will have a significant impact in overall device battery life.

Before we introduce the stationary position collection scheme let usfi rst define how a such a state is detected. In order to detect a *stationary* state we use measurements from both the accelerometer and the gyroscope. These measurements are gathered using a sliding window w of s seconds, which we calculate the standard deviation σ_a and σ_g for the accelerometer and gyroscope, respectively. Additionally, we also keep track their global minimum values as $\sigma_{a,min}$ and $\sigma_{g,min}$.

We define that a stationary state cannot be smaller than 1 second, which helps us eliminate "short"-lived outliers. We consider a position to be stationary if $\sigma_a < 2\sigma_{a,min}$ and $\sigma_g < 3\sigma_{g,min}$ over the duration of a given window. If a window satisfies both conditions we classify it as a valid *stationary* state and store the averaged values for both accelerometer and gyroscope for use in our calibration procedure; examples of such windows can be seen in Figure 4 where we plot both accelerometer and gyroscope traces and mark each window that is deemed to be *stationary*; in this case the detector is configured for continuous monitoring using variable sized windows. It has to be noted that in order to avoid getting data from the same orientation "nearby" states are merged (as this happens in Figure 4. For completeness, a rough sketch of the algorithm used is presented in Algorithm 1 which is shown below; as we can see it allows tweaking many parameters related on how stationary states are detected. Furthermore, we allow to have both continuous and a somewhat stochastic way of when to probe for further measurements and a way to abandon gathering fast in case we cannotfi nd potential candidate without gathering the full amount of measurements required; this is accomplished by using a value for w_{min} as a cut-off point that is set much less than w_{max}. Finally, it has to be noted that this state detection algorithm is *only* executed when needed (e.g. when we need to re-calibrate). The need for further calibration is checked periodically and is based on the quality of the IMU measurements.

4.5 Assisted Gravity Estimation

The primary ground truth that our algorithms use in order to perform the calibration is the force of Earth's *gravity g*, which is usually referenced to have a value of 9.80665 m/s^2. Unfortunately, this is not always the case as gravity force changes based on *location* and *altitude*.

Given this fact, it would be naive and erroneous to use "just" the default value of g. Instead we use location information by probing (once) the GPS of the phone in order to get an accurate location reading as well as the Internet, if available, to also get an altitude estimation by querying online mapping services (e.g. Google Maps).

Algorithm 1 Stationary state detection pseudocode

$s \leftarrow$ number of states to gather
states $\leftarrow$ [] ▷ Empty array for state storage
$\sigma_{a_{\min}} \leftarrow g \pm v$ ▷ Initialize to value close to g
$\sigma_{g_{\min}} \leftarrow 0 \pm v'$ ▷ Initialize to near 0
$p \sim [1 - n]$ ▷ set time to probe IMU from range
$w_{\min} \leftarrow$ min_size ▷ minimum window size
$w_{\max} \leftarrow$ max_size $\sim [w_{\min}, w_{\min} + b]$ ▷ maximum window size
repeat
  if time = p **then** ▷ time reached, probe IMU
    $cw_a, cw_g \leftarrow$ gather $w_{\min}$ number of IMU measurements
    $\sigma_{cw_a} \leftarrow$ calc_std(L2(cw_a)) ▷ calculate std for accel. L_2 norm
    $\sigma_{cw_g} \leftarrow$ calc_std(cw_g) ▷ calculate std for gyro.
    if $\sigma_{cw_a} < 2\sigma_a$ and $\sigma_{cw_g} < 3\sigma_g$ **then** ▷ continue gathering?
      $cw_a, cw_g \leftarrow$ gather $w_{\max}$ number of IMU measurements
      $\sigma_{cw_a} \leftarrow$ calc_std(L2(cw_a)) ▷ update std for accel. L_2 norm
      $\sigma_{cw_g} \leftarrow$ calc_std(cw_g) ▷ update std for gyro.
      if $\sigma_{cw_a} < 2\sigma_{a_{\min}}$ and $\sigma_{cw_g} < 3\sigma_{g_{\min}}$ **then**
        if $\sigma_{cw_a} < \sigma_{a_{\min}}$ **then** ▷ keep track of $\sigma_{a_{\min}}$
          $\sigma_{a_{\min}} \leftarrow \sigma_{cw_a}$
        end if
        if $\sigma_{cw_g} < \sigma_{g_{\min}}$ **then** ▷ keep track of $\sigma_{g_{\min}}$
          $\sigma_{g_{\min}} \leftarrow \sigma_{cw_g}$
        end if
        $w_{\max} \leftarrow$ max_size $\sim [w_{\min}, w_{\min} + b]$ ▷ reassign max size
        states $\leftarrow$ states + [avg(cw_a), avg(cw_g)] as stationary state
      end if
    end if
  end if
  $p \sim [1 - n]$ ▷ re-set time to probe IMU from range
until states.size reaches s

Using these information and the International Gravity Formula (IGF) along with Free Air Correction (FAC) [15] we are be able to calculate a more accurate value of g, g_{local} using the following formula:

$$g_{local} = IGF + FAC \tag{16}$$

Where the IGF is defined as:

$$IGF = 9.780327(1 + 0.0053024sin^2\Phi - 0.0000058sin^2 2\Phi) \tag{17}$$

And FAC is defined as:

$$FAC = -3.086 \times 10^{-6} \times h \tag{18}$$

Where g_{local} is the calculated *local* gravity, Φ is the current device latitude and h is the current device height relative to sea level. This in turn helps us to achieve better calibration results as the value of g, which our methods depend for calibration, is more accurately estimated. Additionally, since the mobile device is bound to be in the

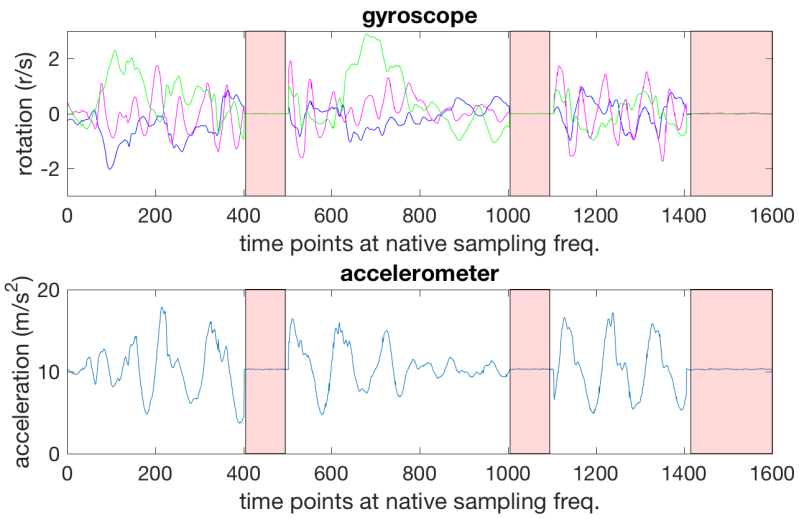


Fig. 4. Detected stationary states

same region for most of the time we do not need to continuously probe location services in order to estimate the g-value. Events that would require us to re-evaluate used value would be changing time-zones, moving to another country and so on. In general though, for validation purposes, we periodically obtain location information but only when it is least expensive, e.g., when the device is charging.

4.6 Electronic Noise Removal

The problem of removing IMU output signal noise has been extensively studied in literature [26, 29]. The most cost-effective way of *reducing* noise to acceptable levels in this particular setting is by using a *median* or a rolling *averaging* filter. In essence bothfi ltering methods provide reasonable results but some characteristics of the averagingfi lter make it a more suitable candidate for our purposes. Concretely, our intuition stems from the the fact that while medianfilter *does not* create *new* values by averaging existing ones it is *less* sensitive to outliers (e.g. sudden spikes) which are a common occurrence in our setting. Additionally, stationary readings tend to be the most extreme ones as far noise as is concerned because it is significantly more evident in such conditions than it would otherwise normally be. Following the above examples, and since the traces are from stationary positions, we can use the *variance* to quantify the performance of the aforementionedfi ltering methods; indicatively the stationary position variances afterfi ltering for both types are shown in Table 1.

Table 1. Signal Output variance

Filter window size	Median Variance	Averaging Variance
10 steps	$4.6495e-05$	$3.4568e-05$
20 steps	$2.6465e-05$	$1.8359e-05$

We can evidently see in Table 1 that the averagingfi lter gives us *better* results when compared to the median. In addition, we also notice that we get better results the larger thefi lter window size gets, which is expected;

although, it has to be noted that increasing thefi lter window size that increases the inherent response *delay* and this variable needs to be balanced against the sampling frequency. Commonly available smartphones have sampling ranges from $1 - 100$Hz with a usual value being, for sensing applications, around 20~30Hz (i.e. 20-30-samples every second). Another, very strong argument for using the averagingfi lter against the median is *power consumption* as medianfi lter is *significantly* more expensive in terms of computational requirements. This stems mainly due to the fact that *sorting* has to be performed at each iteration; whereas the rolling averagefi lter is much more efficient in terms of computational requirements. Finally, the kernel size has to be adjusted based on the sampling frequency in order to have optimal performance; suitable kernel size ranges are 3-20 elements and are roughly assigned as shown in Table 2 below.

Table 2. Frequency steps assignment

Filter window size	Frequency range
3 steps	$1 - 20$Hz
5 steps	$21 - 80$Hz
10 steps	$80 - 120$Hz
20 steps	> 120Hz

4.7 Periodic CalibrationQu ality Check

As we said previously, our scheme is adaptive and thus we have developed a mechanism in order to evaluate the need for re-calibration. Performing re-calibration is very important and after a while, almost mandatory but there has been no "golden-standard" on how frequently it should be performed [20, 40]. Thankfully, checking for calibration quality is much faster and easier than trying to gather potential states that would be suitable for the calibration procedure. Concretely, evaluating the accelerometer calibration quality can be checked at any point by measuring the deviation of the L_2 against our current estimation of gravity acceleration g; on the other hand, fully evaluating the gyroscope calibration quality relies on the phone being in a stationary position. Currently, our method relies on the accelerometer to decide when it is applicable to perform a full re-calibration procedure that calibrates both the accelerometer as well as the gyroscope using the scheme described above; this check happens hourly. Although calibration deterministic error parameters do not change drastically over time for both the gyroscope as well as the accelerometer, it is a known fact that the gyroscope has a constant drift [28]. This can be a significant problem, depending on its quality but thankfully can be easily alleviated in a power efficient way by zeroing gyroscope often, which in our case is performed every time we check for the accelerometer calibration quality.

5 EVALUATION

In this section we evaluate the performance of our auto-calibration scheme and give evidence of how it can help achieve a more stable IMU signal acquisition and therefore boosting classification accuracy, especially in the presence of significant trace similarity. For the study of performance we use a dataset released with paper [36] containing traces of accelerometer and gyroscope from multiple users and devices during different activities. At a high level the evaluation is composed of these parts:

- We report the effect of the using calibration on the resulted homogeneity of the activity sensor traces from different devices
- We evaluate the benefit of the calibration also over the classification using advanced neural network classifiers, as well as simpler SVM ones, especially in the the case of *significantly* similar activities.

- We study the calibration quality improvement when using a better inference of g (as described in Section 4.5).
- We estimate the energy footprint our technique, which is shown to be significantly lower than Kalman-filtering techniques.
- We perform a running analysis on real-world usage in order to evaluate the cost of our method so that we can better quantify the energy/accuracy trade-off that performing calibration can impose.

The results of the evaluation show that our approach offers an improvement of about $2 \sim 5\%$, on average, over the classification error when using a calibrated input for a neural network classifier and $9 \sim 12$ improvement when using a reference SVM classifier. It is important to note that this improvement enables much more precise labelling of activities and potentially unlocks the level of precision needed by health applications to be trusted more widely. Finally, it has to be noted that *unlike* previous calibration schemes similar to ours they are *one-shot* calibration techniques which require user-intervention for performing both the *initial* or *subsequent* calibrations (if supported) (e.g. [13]); thus, direct comparison with our scheme is not directly applicable.

5.1 Dataset and Pre-processing

We now describe the details of the data used in the section. We use a dataset released alongside paper [36] which has records of accelerometer and gyroscope traces from multiple users and devices while performing different activities. This dataset includes activities that have very similar traces (e.g. climbing stairs up and down). We will show how calibration is vital for the distinction of these cases, with improved accuracy.

The aforementioned dataset is comprised out of traces from 9 users using 8 smartphones and 4 smartwatches performing 5 different activities. The performed activities were *scripted*; which means that all 9 users performed a *specified* routine for each activity which included the following: *walking*, *walking stairs up*, *walking stairs down*, *sitting*, and *biking*. Each participant while performing each scripted activity was strapped with a tight pouch located around his waist which contained all 8 smartphones and wore on each arm 2 smartwatches. All activities that the participants conducted lasted 5 minutes which ensured an equal data distribution amongst the given activity classes. All the recordings were done using the *highest* available sampling frequency of each device and the generated tuple was labeled by the respective sensor values along with the emit time-stamp which the operating system attaches; whereas the activity type, was embedded to each tuple by the authors. The authors took into consideration, in order to minimize external factors affecting the data acquisition, to keep the CPU load of each device minimal having active only the data acquisition app. Furthermore, each activity had two different environment and routes in which the activity was to be performed; the participants then were divided into two groups, each of which used the *same* environment and routes for the execution of their *scripted* activities. Unfortunately, for the activity recognition experiments only a *subset* of the devices they used in [36] is employed; which, for the sake of completeness, are shown below in Table 3.

Table 3. Devices included in the publicly available activity dataset

	#	Release	Max. sampling rate (Hz)
Smartwatch			
LG G	2	2014	200
Samsung Galaxy Gear	2	2013	100
Smartphone			
LG Nexus 4	2	2012	200
Samsung Galaxy S3	2	2012	150
Samsung Galaxy S3 Mini	2	2012	100
Samsung Galaxy S+	2	2011	50

For our evaluation we *only* use the traces of the included *smartphones* as our method currently does not support *smartwatches*. Before we proceed, we need tofi rst define what is a *similar activity*.

Definition 5.1. The similarity of an activity against another is defined as the summation of the Mean Squared Error (MSE) of the original traces and the MSE of the L_2 norm calculated against another activity over the same period of time.

For our dataset the most similar activity pairs based on the similarity metric defined above are shown in Table 4. We set the cutoff threshold for a similarity pair to be labeled as significantly similar if for activities a, b, $Sim(a, b) < 0.25$. Each category of activity traces was aggregated over all smartphone devices and users and was normalized before passing it to the similarity function to calculate thefi nal value.

Table 4. Similarity metric to similar activities

Activity Pair	Activity Similarity
Walking vs Stairs up	0.219290
Walking vs Stairs down	0.178262
Stairs up vs down	0.210257
Walking vs Biking	0.223994

These values were calculated by taking the similarity metric for all activities over our dataset and comparing their values. From the data, we also generated a synthetic trace of sequentially ordered activities which we use to train our calibration algorithm. This sequence is produced by using slices of activities in the dataset and stitching them together. The sequence is designed to have 12-20 suitable stationary states (sitting or idle) while other types of activities (walking, climbing stairs, biking) are interleaved in between. Each slice is composed of 500 measurements making the total length of the training sequence of 6000-10000 measurements.

5.2 Results

The evaluation is conducted in four parts. Firstly we test the homogeneity of the activity traces, for both uncalibrated and calibrated measurements, as one of our principal target is to have more consistent traces across different devices. Secondly we evaluated the benefit of having a calibrated signal instead of a uncalibrated one in a typical activity classification problem, which involved training a neural network as well as an SVM classifier and measuring the potential classification improvement. We then evaluate the benefit in terms of calibration quality when using a better estimate of g as inferred by exploiting information obtained probing (sporadically) the GPS services (thus giving us a good estimation of the location of the device) as described in Section 4.5. Concluding our results, in Section 5.3 we evaluate the energy efficiency of our method which shows a three times lower footprint compared to Kalman-filtering based methods.

Calibration Effect on Trace Homogeneity. The purpose of this test is to evaluate if calibration offers any tangible benefit in making traces from different devices more homogeneous. To that end, we created a test to measure how similar different signals are against a global representation for uncalibrated and calibrated traces, which we call a *normalized aggregate* trace defined below.

Definition 5.2. We define as a *normalized aggregate* trace the product of adding n traces over the same period of time and normalizing the result by n.

Concretely, an example of such a trace would one that is shown in Figure 5; which shows the traces fromfi ve users while walking when using an LG Nexus 4 phone along with its aggregated trace (drawn in pastel blue) as

defined in Definition 5.2. The principal intuition behind this is to get a *general* and an *approximate* view on how that particular activity trace would look from that specific device; in addition, the more *similar* these traces are to the *normalized aggregate* then the more *homogeneous* these traces will be (i.e. *similar* to each other). Obviously, this principle could be expanded to generate *normalized aggregates* per *activity* for *all* devices but for readability purposes we limited the aggregation presented to just 5 traces.

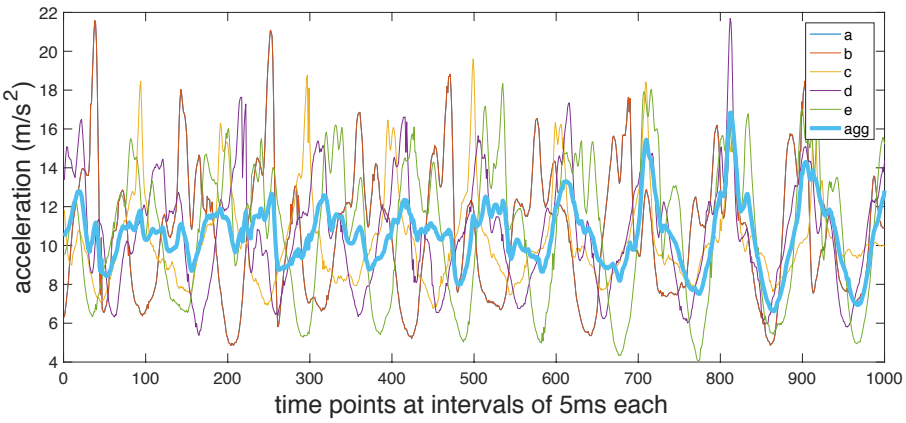


Fig. 5. Aggregated trace from 5 users walking trace when using a Nexus 4 phone

Then we take the *normalized aggregate* trace and measure how similar it is against the signals used to produce it, using the Mean Squared Error [39] (from now on MSE), essentially measuring how similar each respective signal is to the *normalized aggregate*. The *lower* the MSE is the *more similar* the respective traces are.

In order to perform the evaluation of how similar the traces are when using calibration or not we compared the MSE of the calibrated and uncalibrated traces. Specifically, we created two *normalized aggregates* for each distinct activity using uncalibrated and calibrated traces. For each uncalibrated trace we calculated the MSE against the respective *normalized aggregate* which was then added and averaged by the number of different traces; the same procedure was performed for the calibrated traces.

Table 5. MSE in Uncalibrated vs Calibrated traces

Activity Type	Uncalib. MSE	Calib. MSE
Walking	0.1011	0.0689
Stairs up	0.1064	0.0552
Stairs down	0.1203	0.0717
Sit	0.0009	0.0009
Biking	0.0186	0.0086

The results shown in Table 5 indicate that the calibrated traces are more similar to each other, as lower MSE against the *normalized aggregate* traces represent a more homogeneous trace collection across devices. This, in turn, tells us that the traces are devoid of a significant amount of variation introduced by sensor imperfections, hardware manufacturing variability, operating system differences among other things.

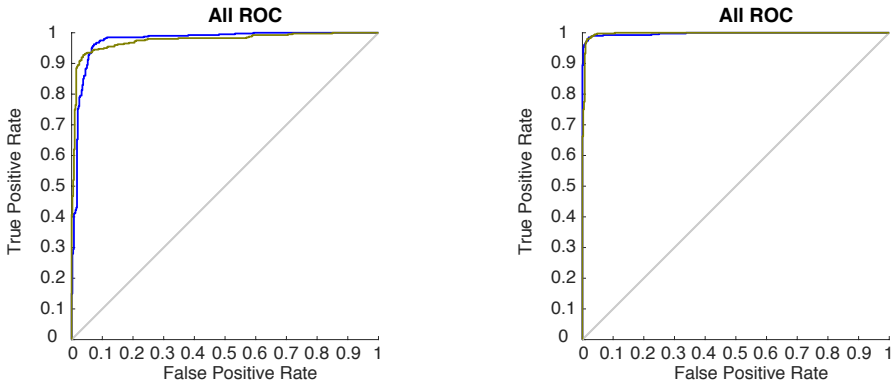
Impact on Classification. For the second task we used two types of classifiers, thefi rst one was a typical neural network classifier [6, 17] using 25, 50 and 100 hidden layers, respectively; while the second one was an SVM binary classifier [19, 32]. Each pattern to be classified was defined as a slice of 40 *consecutive* measurements which were then fed to train the three different neural networks and the SVM classifier. The reasoning behind the usage of both is that a neural network classifier is quite resilient against noisy data and thus would be indicative of a lower bound in classification improvement; while the SVM classifier generally being lower performing when compared to neural network was selected to quantify the benefit when using a less ideal classifier. Additionally, we performed the classification for *similar* traces, as defined in 5.1 in order to evaluate the classification accuracy when the actual traces are *significantly* similar to each other. In our current dataset such activity signals were: *walking*, *biking climbing stairs up* and, *climbing stairs down*. The intuition behind this is that because the essential task of a classifier is to distinguish amongst different classes, then the more distinct those classes are the easier it will be for the classifier to perform the classification; thus having cleaner, more robust data by using calibration should provide a tangible classification improvement. The gain range in classification accuracy for the SVM and all three network configurations when classifying using uncalibrated and calibrated data is shown in Table 6.

Table 6. Classification boost over non-calibrated readings

Activity Type	25 HL	50 HL	100 HL	SVM
stairs up vs walk	$\sim 2-5\%$	$\sim 2-5\%$	$\sim 4-5\%$	$\sim 9\%$
stairs down vs walk	$\sim 2-4.5\%$	$\sim 2-4.5\%$	$\sim 4-5\%$	$\sim 8\%$
stairs down vs up	$\sim 5-10\%$	$\sim 5-8\%$	$\sim 6-9\%$	$\sim 9\%$
walking vs biking	$\sim 4-8\%$	$\sim 4-7\%$	$\sim 3-6\%$	$\sim 12\%$

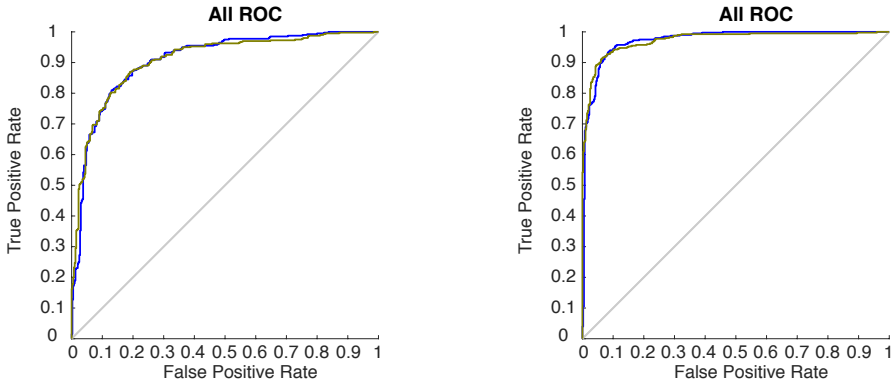
Starting with a simpler classifier using SVM we observed that the average baseline was worse when compared to more advanced classifiers like neural networks, which are much more resilient to noise. In addition, we observed a overall classification boost ranging from $9-12\%$ from the baseline SVM classification which was around $\sim 80\%$. This in-turn brought the classification accuracy to around 90% which is a considerable improvement when compared to the baseline. Moving on to the use of neural networks as a classifier it is a known fact that they are quite resilient when dealing with noisy data [14], hence even in the setting of uncalibrated traces achieved pretty high classification accuracy; this is also in line with results produced for activity detection in [36] albeit using different training features. Concretely, when classifying walking against climbing stairs the classification accuracy with uncalibrated data is around 92%, but as we see from the above table using calibrated traces we were able to add a $2 \sim 5\%$ improvement in classification accuracy. Interestingly enough, when we try to classify even one of the more similar traces, specifically climbing stairs upwards and downwards we obtain a larger improvement in classification accuracy when compared to walking. This is because neural networks achieved around 85% accuracy when classifying between these two activities climbing stairs upwards and downwards; we observe a similar behaviour when classifying walking and biking activities. These results show that using calibrated traces boosted classification accuracy by $5 \sim 10\%$ which can be quite significant when considering activity the context of the activity and medical applications. It is in these cases where it is really important to distinguish with high precision, for example, if someone is just walking or biking. It might just be that the improvement of accuracy within these activities boundaries is what makes mobile sensing applications trustworthy as it allows to achieve near perfect classification accuracy ($> 95\%$). For brevity, indicative ROC curves for Walking vs Climbing stairs up and Climbing stairs upwards or downwards for the 100 HL neural network configuration are shown in Figure 6 and Figure 7, respectively. Thesefi gures show a quantifiable improvement in classification when using state of the art classifiers especially in the case climbing stairs up or down where from $\sim 85\%$ it raises to $\sim 94\%$.

Finally, it has to be noted that in some cases as is shown in Table 6 when classifying using neural networks employing more hidden layers results in worse classification performance. Unfortunately, this is the result of our neural networking *overfitting* and this is well known and studied problem in literature [18, 35]. Overfitting, usually is dependent on the size of the training dataset and the amount of hidden layers present; in our case and, as results show, given our dataset size it seems that going past 25 hidden layers results our neural network overfitting the data. This is a classic example of *more* resources are not always *better*.



(a) uncalib. 100 HL (b) calib. 100 HL

Fig. 6. Walking vs Climbing stairs up ROC



(a) uncalib. 100 HL (b) calib. 100 HL

Fig. 7. Climbing stairs down vs up ROC

Classification Comparison Against a reference Kalman filter implementation. In order to evaluate our method against a widely used method for performing an adaptive correction scheme like ours, we implemented a reference Kalmanfi lter for comparison. We used the same values for all three axes for the disturbance and sensor

noise variance, usually referenced as Q and R respectively. Additionally, two separatefi lters were used for the Accelerometer and Gyroscope that were independent of each other. The combined results, for both methods, are shown in Table 7.

Table 7. Classification boost over non-calibrated readings

Activity Type	25 HL	50 HL	100 HL	SVM
Presented Method				
stairs up vs walk	$\sim 2-5\%$	$\sim 2-5\%$	$\sim 4-5\%$	$\sim 9\%$
stairs down vs walk	$\sim 2-4.5\%$	$\sim 2-4.5\%$	$\sim 4-5\%$	$\sim 8\%$
stairs down vs up	$\sim 5-10\%$	$\sim 5-8\%$	$\sim 6-9\%$	$\sim 9\%$
walking vs biking	$\sim 4-8\%$	$\sim 4-7\%$	$\sim 3-6\%$	$\sim 12\%$
Kalmanfilter				
stairs up vs walk	$\sim 0.5-2\%$	$\sim 1-3\%$	$\sim 2-3\%$	$\sim 1\%$
stairs down vs walk	$\sim 1-1.5\%$	$\sim 1-2.5\%$	$\sim 2-2.5\%$	$\sim 0.5\%$
stairs down vs up	$\sim 1-2\%$	$\sim 2.5-4\%$	$\sim 3-4\%$	$\sim 2\%$
walking vs biking	$\sim 2-4\%$	$\sim 3-4.5\%$	$\sim 2-3.5\%$	$\sim 2\%$

From the presented results we see that our scheme consistently outperforms the basic Kalman-filter based approach, in terms of classification. Theoretically, a Kalmanfi lter is a linear estimator and has been proven to be optimal; in our case it helped remove the inherent noise as well as provide a more stable smoothing result without the need of a separatefi ltering method but failed to correctly account for the inherent sensor bias that is present. This meant that although we got stable results, as far as, smoothing and signal stability was concerned the inherent bias heterogeneity that each device has was still present when using our reference implementation. This results in some improvement in classification when not using a correction scheme at all, but due to the fact that it was not able to completely account for the inherent sensor bias itself the improvement was small. In fact, this is also noted in literature and some references have been made in combining offline calibration methods with Kalmanfi lters to achieve the optimal solution (e.g. [3]). Unfortunately, the scheme proposed in [3] although being able to provide an estimation from thefi rst iteration it is quite expensive power-wise, mandates the usage of specialized precision equipment (a robotic arm) and requires quite a few iterations to converge. Obviously, these constraints make this scheme not suitable for use in a mobile setting.

Calibration over gravity variations. We further evaluated the calibration results when using the default g value against the one we obtain from using our assisted gravity estimation technique described in Section 4.5. To create some reasonable test dataset we created synthetic traces byfi rst normalizing the traces by g and then multiplying by the target g-value; we then used our algorithm to calibrate using as input the default g value and the estimated one. The results in Table 8 show the calibration residual error in cost function E when using the default g ($g = 9.80665m/s^2$) value and the local g value estimation in a specific location ($g = 9.81676m/s^2$).

Table 8. Calibration performance using default and improved g estimation

Component	default-g	est.-g
Accelerometer	0.6015	0.0090
Gyroscope	0.2015	0.1276

As we can see in Table 8 assisted gravity estimation gives us a tangible improvement in overall calibration quality when compared to just using afi xed value for gravity acceleration. Inferring the estimated g value is performed by fusing information obtained from mobile device's location services as indicated in Section 4.5.

5.3 Energy Consumption Analysis

Having our method to be applicable to most mobile devices, the computation footprint should be as low as possible, For this reason, we elected to use the averagingfi lter against the median (as at every iteration sorting has to be performed, see Section 4.6). Energy-wise, what consumes the largest amount of power is the initial calibration procedure because the full number of stationary states have to be detected and stored. This can be done in two ways, thefi rst one is start and gathering traces immediately and the other is to use traces from within the application when it is being used but that has the downside of potentially not being able to gather the required amount of states as quickly. Preferably, initial calibration procedure should be performed, after collecting at least 9 stationary states and calculate the parameters when the device is charging; then results can be used to perform the correction against the uncalibrated measurements; this operation needs to happen for every uncalibrated measurement received. For practical purposes the amount of operations needed to perform is adding two 3×1 vectors and multiplying the result with a 3×3. This requires, in total, 9 additions (3 from vector addition and 6 from multiplication) and 9 multiplications which have to be performed two times; one for the accelerometer and one for the gyroscope. In practice we have an insignificant power cost, at least when compared to the power the IMU draws by itself when active.

5.3.1 Power Efficiency Comparison Against a Kalman filter implementation. In order to further evaluate our claims we compared our implementation against a Kalman-filtering based sensor fusion scheme for the gyroscope and the accelerometer, based on freely available libraries. Let usfi rst also consider that Kalman-filter has a *quadratic* ($O(n^2)$) complexity cost for the *update* step and an $O(n^3)$ total cost [33] which is quite expensive, computationally wise and as we will show power-wise as well. Our target was to measure the power consumption using a high polling rate of these two methods. The measures were based on a Samsung Galaxy S7 running Android 7.0. To perform our tests all radios were disabled and all active applications were removed, then a baseline was obtained which was around 165mA (with screen on). Following the acquisition of a stable baseline we executed each method using the second highest polling rate available (`SENSOR_DELAY_GAME`) and measured the power consumption around every 5 seconds until we obtained 20 measurements for each method. Before reporting, we averaged the measurements, subtracted the baseline and rounded to the nearest .5; this was done in order to better reflect the energy footprint of each method. The results of our testing are presented in Table 9.

Table 9. Power consumption measurements

Method	Power (avg)
Accelerometer (Kalman)	191.00mA
Accelerometer	57.00mA
Gyroscope (Kalman)	186.50mA
Gyroscope	59.5mA
Combined (Kalman)	220.50mA
Combined	68.5mA

We used a high polling value in our experiment (20 ms polling as per Android 7.0 documentation) most of the times much higher than in any real world applications, especially ones designed to run in the background for long periods of time. Additionally, we only measured what needs to be computed at every iteration; this does not

include our calibration procedure (as this can happen when the device is charging) but only its application. As we can see from our presented results our method can provide significant energy savings, even at high polling rates when compared to an indicative Kalman-filter based method.

5.3.2 Actual usage power analysis. To further evaluate the power cost of calibration in general and its potential power/efficiency trade-off we performed the following experiment using real-world data. We created an sample application which would run in the background while the phone was powered on, while constantly polling the IMU at 25Hz in the background; this was performed while running our calibration algorithm and without. During the experiment the phone was located in a pocket in one of the authors and was carried around during the day, which would simulate real-world usage of the phone. We feel this is a perfect way of measuring the actual cost as it is devoid of any overhead any additional components that are commonly present in more complex applications (e.g. on device HAR classifiers, games, and so on). If present, these complex segments impose a significant power cost on the application making hard to distinguish the actual power overhead of calibration. In order to measure the power consumption over time and how often calibration needed to be performed we used the slope of the battery discharge curve which is provided by the operating system in order to calculate the estimated power consumption. Although the measurement is not performed using a hardware monitor (like monsoon) as it is shown in [8] it comes incredibly close and for our intents and purposes this will suffice; this is the case because we require the device to be movable, which makes it impossible to use a complex setup while using the device.

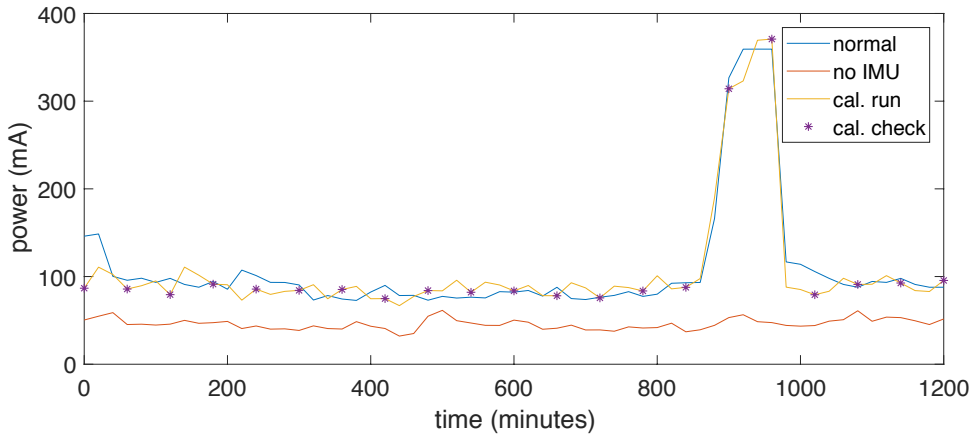


Fig. 8. Total device power usage over time

In Figure 8 we can see the traces gathered in order to evaluate the calibration actual energy cost; in both cases the IMU was being probed constantly in the background; most of the time the difference in power is very small. The huge spike seen around 870-950, is in order to evaluate the energy usage while using an intensive application we watched about 70 minutes of videos through the YouTube app. This experiment showed that the battery life impact calibration imposes is expected to be minimal, even when a higher load is present. Finally, we have a control signal of the same device that does not have its IMU probed for the same amount of time in order to see the power consumption difference.

In addition, unless a significant geographical change happens, season change (due to temperature) or enough time passes the deterministic errors stay relatively the same thus maintaining the calibration quality over long

periods of time [30]. If such a change occurs, and an significant quality degradation is detected then the full calibration takes place which happens after enough stationary states have been gathered. This process takes less than 10 seconds to run and consumes about 120mA-150mA during execution. Of course, we can schedule the calibration to be executed when the device is charging and thus negating the small potential penalty of this cost completely.

6 DISCUSSION

Our approach represents a step forward for calibration methods in mobile devices since it allows the procedure to be completed user unaided. It will also periodically check calibration quality and adjust the model parameters accordingly in order to keep sensor output quality high. Moreover, thefl exible model used can be adopted for use in other sensors such as magnetometer, level, barometer and others. Additionally, the reference g used by the calibration procedure in our technique is more accurate as it is approximated by inferring the value using location data provided by the phone instead of just a static default value. To our knowledge this is thefi rst method specifically designed for mobile devices which also has a quantified energy/accuracy penalty that our evaluation showed it is quite low. Of course, this result was expected as the cost of using the IMU is much higher when compared to the amount of processing power that our scheme requires.

These features provide an activity classification boost which might pave the way for more reliable or more granular detection of activities. Examples of more distinguishable classes might include regular walking versus "power"-walking or fast stairs climbing versus regular climbing. The ability to add precision in classification of these boundary cases in power efficient way is the key to unlock the next generation of continuous mobile sensing applications which can offer trustworthy interpretations of human behavior. Currently, our calibration procedure has been designed and tested on regular smartphones but as wearables (e.g. smartwatches) are becoming more and more common it would be logical to adapt our technique to be used in such devices. Although we have not tried to evaluate against such data adaptation to wearable devices is straightforward, methodology wise, but it should be carefully implement due to the inherent strict computational and power constraints these devices have. Another interesting application of our technique is in the domain of security. As calibration is designed to remove the uniqueness or biases that might exist in different sensors, our results may be used to mitigate the bias-variance deanonymization techniques that have been recently introduced and studied in [4, 7, 25].

Limitations of our method include the fact that currently it does not take in account parameters required by high-end IMUs (e.g. tactical grade ones like [1]), such as temperature bias coefficients, specific sensor noise deviation, absolute limits expected values; realistically speaking though most these parameters are not accessible through the operating system and those that are require rooting of the device. Moreover, our approach is dependent on the number of *distinct* stationary positions we can detect over a period of time; as such, the exact time needed to calibrate cannot be guaranteed. This happens due to the fact that each particular user is expected to have different usage patterns which might affect the number of states we can detect over the same period of time. Although, if normal usage is expected, we can gather enough states to perform our calibration within one day.

7 CONCLUSION AND FUTURE WORK

In this paper we have introduced a novel unaided adaptive auto-calibration method for use in mobile devices. From our evaluation we deduce that using calibrated signals can provide a significant boost in classification accuracy. Our evaluation shows that our method can provide a tangible improvement in classification accuracy while using state of the art neural network classifiers, with accuracy scores to be over 95% and when using simpler SVM classifiers close to 90%. Our use of uncalibrated signals shows that there are corner cases in which classification accuracy drops below 90%. Moreover, our experiments show that the energy/accuracy trade-off

when using calibration is quite low and thus allows it to be used more widely. Concretely, we think our proposed calibration scheme strikes a nice balance between accuracy, power consumption, and ease of use especially in the mobile setting; we can recommend our calibration scheme for usage in cases where we require sufficiently accurate calibration (not military grade), while having a very low overhead and being completely transparent for the end-user both for the initial calibration as well as parameter maintenance. Finally, we plan to now explore the potential of the technique in continuous mobile sensing applications with a particular focus on m-health, attempting deployments with clinicians and patients involved.

ACKNOWLEDGMENTS

This work was supported by The Alan Turing Institute under grants: TU/C/000003, TU/B/000069, and EP/N510129/1. Finally, we would like to deeply thank all the anonymous reviewers who dedicated their time providing instrumental and constructive comments which greatly helped us to improve the quality of this submission.

REFERENCES

[1] Brochure for analog devices tactical grade IMU. http://www.analog.com/media/en/news-marketing-collateral/product-highlight/Tactical-Grade-IMU.PDF. Accessed: 10-04-2017.

[2] P Aggarwal, Z Syed, X Niu, and N El-Sheimy. A standard testing and calibration procedure for low cost mems inertial sensors and units. *The Journal of Navigation*, 61(2):323–336, 2008.

[3] Tadej Beravs, Janez Podobnik, and Marko Munih. Three-axial accelerometer calibration using kalmanfi lter covariance matrix for online estimation of optimal sensor orientation. *IEEE Transactions on Instrumentation and Measurement*, 61(9):2501–2511, 2012.

[4] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. Mobile device identification via sensorfingerprinting. *arXiv preprint arXiv:1408.1416*, 2014.

[5] Francois Caron, Emmanuel Duflos, Denis Pomorski, and Philippe Vanheeghe. Gps/imu data fusion using multisensor kalmanfiltering: introduction of contextual aspects. *Information fusion*, 7(2):221–230, 2006.

[6] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[7] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *NDSS*, 2014.

[8] Mian Dong and Lin Zhong. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 335–348. ACM, 2011.

[9] Mohammed El-Diasty and Spiros Pagiatakis. Calibration and stochastic modelling of inertial navigation sensor errors. *Journal of Global Positioning Systems*, 7(2):170–182, 2008.

[10] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. Analysis and modeling of inertial sensors using allan variance. *IEEE Transactions on instrumentation and measurement*, 57(1):140–149, 2008.

[11] Franco Ferraris, Ugo Grimaldi, and Marco Parvis. Procedure for effortless in-field calibration of three-axial rate gyro and accelerometers. *Sensors and Materials*, 7(5):311–330, 1995.

[12] WT Fong, SK Ong, and AYC Nee. Methods for in-field user calibration of an inertial measurement unit without external equipment. *Measurement Science and technology*, 19(8):085202, 2008.

[13] Iuri Frosio, Federico Pedersini, and N Alberto Borghese. Autocalibration of mems accelerometers. *IEEE Transactions on Instrumentation and Measurement*, 58(6):2034–2041, 2009.

[14] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[15] William J Hinze, Ralph RB Von Frese, and Afif H Saad. *Gravity and magnetic exploration: Principles, practices, and applications*. Cambridge University Press, 2013.

[16] Christopher Jekeli. *Inertial navigation systems with geodetic applications*. Walter de Gruyter, 2001.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Steve Lawrence, C Lee Giles, and Ah Chung Tsoi. Lessons in neural network training: Overfitting may be harder than expected. In *AAAI/IAAI*, pages 540–545, 1997.

[19] Yi Liu and Yuan F Zheng. One-against-all multi-class svm classification using reliability measures. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 849–854. IEEE, 2005.

[20] Joost Conrad Lötters, J Schipper, PH Veltink, W Olthuis, and P Bergveld. Procedure for in-use calibration of triaxial accelerometers in medical applications. *Sensors and Actuators A: Physical*, 68(1-3):221–228, 1998.
[21] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM conference on embedded networked sensor systems*, pages 71–84. ACM, 2010.
[22] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. Methods for non-linear least squares problems. 2004.
[23] Andrea Mannini, Stephen S Intille, Mary Rosenberger, Angelo M Sabatini, and William Haskell. Activity recognition using a single accelerometer placed at the wrist or ankle. *Medicine and science in sports and exercise*, 45(11):2193, 2013.
[24] J Metge, R Mégret, A Giremus, Y Berthoumieu, and T Décamps. Calibration of an inertial-magnetic measurement unit without external equipment, in the presence of dynamic magnetic disturbances. *Measurement Science and Technology*, 25(12):125106, 2014.
[25] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *USENIX Security*, pages 1053–1067, 2014.
[26] T Mineta, S Kobayashi, Y Watanabe, S Kanauchi, I Nakagawa, E Suganuma, and M Esashi. Three-axis capacitive accelerometer with uniform axial sensitivities. *Journal of Micromechanics and Microengineering*, 6(4):431, 1996.
[27] Faraz M Mirzaei and Stergios I Roumeliotis. A kalmanfi lter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation. *IEEE transactions on robotics*, 24(5):1143–1156, 2008.
[28] Howard Musoff and Jerold P Gilmore. Inertial navigation system with automatic redundancy and dynamic compensation of gyroscope drift error, March 16 1993. US Patent 5,194,872.
[29] K Nirmal, AG Sreejith, Joice Mathew, Mayuresh Sarpotdar, Ambily Suresh, Ajin Prakash, Margarita Safonova, and Jayant Murthy. Noise modeling and analysis of an imu-based attitude sensor: improvement of performance byfi ltering and sensor fusion. In *SPIE Astronomical Telescopes+ Instrumentation*, pages 99126W–99126W. International Society for Optics and Photonics, 2016.
[30] Shashi Poddar, Vipan Kumar, and Amod Kumar. A comprehensive overview of inertial sensor calibration techniques. *Journal of Dynamic Systems, Measurement, and Control*, 139(1):011006, 2017.
[31] Vijay Raghunathan, Curt Schurgers, Sung Park, and Mani B Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal processing magazine*, 19(2):40–50, 2002.
[32] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.
[33] Saiful Bahri Samsuri, Hairi Zamzuri, Mohd Azizi Abdul Rahman, Saiful Amri Mazlan, and Abdul Hadi Abd Rahman. Computational cost analysis of extended kalmanfi lter in simultaneous localization and mapping (ekf-slam) problem for autonomous vehicle. *ARPN Journal of Engineering and Applied Sciences*, 10(17):153–158, 2015.
[34] Isaac Skog and Peter Händel. Calibration of a mems inertial measurement unit. In *XVII IMEKO World Congress*, pages 1–6, 2006.
[35] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.
[36] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 127–140. ACM, 2015.
[37] Salah Sukkarieh, Eduardo Mario Nebot, and Hugh F Durrant-Whyte. A high integrity imu/gps navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation*, 15(3):572–578, 1999.
[38] ZF Syed, P Aggarwal, C Goodall, X Niu, and N El-Sheimy. A new multi-position calibration method for mems inertial navigation systems. *Measurement Science and Technology*, 18(7):1897, 2007.
[39] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signalfi delity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.
[40] Martin L Wilson. Recommended calibration interval. 2010.
[41] Seong-hoon Peter Won and Farid Golnaraghi. A triaxial accelerometer calibration method using a mathematical model. *IEEE Transactions on Instrumentation and Measurement*, 59(8):2144–2153, 2010.
[42] Pifu Zhang, Jason Gu, Evangelos E Milios, and Peter Huynh. Navigation with imu/gps/digital compass with unscented kalmanfi lter. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 3, pages 1497–1502. IEEE, 2005.

A ANALYTICAL DERIVATION OF ACCELEROMETER FORMULA

Let us start this derivation analysis byfi rst mathematically expressing Definition 4.1 as is shown in Equation 19.

$$\sqrt{x^2 + y^2 + z^2} = g \tag{19}$$

In addition to the previous definition, when the accelerometer *is* moving, the following holds.

Definition A.1. When the *ideal* accelerometer *is* moving its acceleration modulus across all of its axes is equal to g (acceleration due to gravity) plus a which is its real acceleration, both measured in m/s^2. □

In essence, this is a more generic version of Equation 19, as when the accelerometer is static a is bound to be equal to zero (0); which again mathematically can be expressed as follows:

$$\sqrt{x^2 + y^2 + z^2} = g + a \tag{20}$$

In order to transform the equation to a least squares problem we can do the following:

$$x^2 + y^2 + z^2 = (g + a)^2 \tag{21}$$

Or even more succinctly:

$$x^2 + y^2 + z^2 - (g + a)^2 = 0 \tag{22}$$

Unfortunately, there is no such thing as an *ideal* accelerometer in practice and all IMU sensors suffer from noise and bias; this in turn, can cause a significant deviation in sensed data from raw values which do not compensate for these errors. This error can be expressed as an additive variable that might differ each time we read the raw values, hence when we are dealing with an *non-ideal* accelerometer its error ϵ at any given time has the following formula:

$$x^2 + y^2 + z^2 - (g + a)^2 = \epsilon \tag{23}$$

The error variable ϵ, can be expanded into *two* components; the error that stems from bias (b_ϵ) and the error that stems from electronic noise (n_ϵ). Therefore the fully expanded equation would be the following:

$$x^2 + y^2 + z^2 - (g + a)^2 = b_\epsilon + n_\epsilon \tag{24}$$

To perform our calibration and to calculate the bias estimates we will *always* assume that the mobile device is in *static* position; hence its *real* acceleration (a) is equal to zero (0). This fact allows us to transform Equation 24 as follows:

$$x^2 + y^2 + z^2 - g^2 = b_\epsilon + n_\epsilon \tag{25}$$

Received August 2017; revised November 2017; accepted January 2018.