

Improved Calibration Procedure for Wireless Inertial Measurement Units without Precision Equipment

Fritz Webering*, Sarah Kleinjohann†, Nils Stanislawski*, Holger Blume*

*Institute of Microelectronic Systems, Leibniz University Hannover, Hannover, Germany

†Student at Leibniz University Hannover, Hannover, Germany

*{webering, stanislawski, blume}@ims.uni-hannover.de

†sarah.kleinjohann@stud.uni-hannover.de

Abstract—Inertial measurement units (IMUs) are used in medical applications for many different purposes. However, an IMU’s measurement accuracy can degrade over time, entailing re-calibration. In their 2014 paper, Tedaldi et al. presented an IMU calibration method that does not require external precision equipment or complex procedures. This allows end-users or personnel without expert knowledge of inertial measurement to re-calibrate the sensors by placing them in several suitable but not precisely defined orientations. In this work, we present several improvements to Tedaldi’s method, both on the algorithmic level and the calibration procedure: adaptations for low noise accelerometers, a calibration helper object, and packet loss compensation for wireless calibration. We applied the modified calibration procedure to our custom-built IMU platform and verified the consistency of results across multiple calibration runs. In order to minimize the time needed for re-calibration, we analyzed how the calibration result accuracy degrades when fewer calibration orientations are used. We found that $N=12$ different orientations are sufficient to achieve a very good calibration, and more orientations yielded only marginal improvements. This is a significant improvement compared to the 37 to 50 orientations recommended by Tedaldi. Thus, we were reduced the time required to calibrate a single IMU from ca. 5 minutes to less than 2 minutes without sacrificing any meaningful calibration accuracy.

Index Terms—inertial measurement unit, calibration, wireless

I. INTRODUCTION

Inertial measurement units (IMUs) are used in a wide variety of medical and sports applications, like injury rehabilitation, training progress monitoring, or fall detection in elderly patients, among many others [1]–[3]. Even more complex tasks such as full-body human motion capture have also been addressed using inertial sensors [4], [5]. In areas where precise measurements and orientation estimates are required, for example in sports or motion capturing, correct calibration of each unit is crucial.

When the parameters of the embedded inertial sensors change over long times or varying temperatures, the unit will no longer adhere to its specified accuracy limits, and a re-calibration becomes necessary. The method proposed by Tedaldi et al. [6] in 2014 allows users to re-calibrate their IMUs in the field without the presence of any specialized and often expensive reference equipment like right angles, turntables, or high precision servo platform [7]. When we implemented this method using our in-house wireless IMU

platform, which is described in Section III, we noticed some problems. After careful examination of the algorithm and associated procedures, we were able to overcome the issues as described in Section IV.

The calibration method presented by Tedaldi et al. uses raw accelerometer and gyroscope data [6] which is processed on a host device. The original implementation is in C++ [8], but for our experiments, we used the MATLAB implementation written by Jianzhu Huai [9] because we found the source code easier to use, understand, and modify.

II. RELATED WORK

Another calibration scheme without external equipment was proposed by Ren et al. [10] in 2015 and is based on the concept of rotating the IMU on an inclined plane, thereby determining the heading using the accelerometer. The underlying principle is fundamentally similar to the concept of Tedaldi’s algorithm, but the calibration protocol is more complex than simply laying the IMU in different random orientations. While Ren et al. reported exceptional calibration accuracy and mentioned Tedaldi’s earlier work, they do not compare the accuracy of the two methods.

Peng et al. [11] presented a similar approach to ours in 2022, who also used a 3D-printed icosahedron to orient the IMU during calibration. They implement a simplified calibration procedure based on an iterative weighted Levenberg–Marquardt algorithm on an embedded microcontroller. However, they disregard axis misalignment and suppose the availability of an expensive precision multi-axis servo stage for gyroscope calibration. This makes the algorithm a bit pointless because that same platform could also calibrate the accelerometer.

III. IMU PLATFORM

In order to have a fully configurable IMU development platform, we developed our own wireless sensor device. The development platform was optimized regarding sensor data accuracy, size, weight, and power consumption.

The System-on-Chip (SoC) CC2652R1 by Texas Instruments was chosen as microcontroller (μC). It includes a 32-bit ARM Cortex M4F core, a 2.4 GHz Bluetooth-Low-Energy (BLE) transceiver powered by an ARM Cortex M0

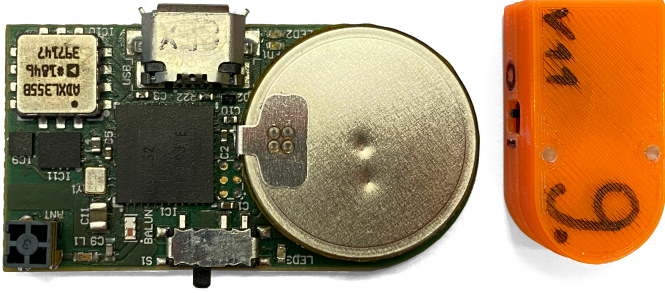


Fig. 1: Left: PCB of the 6th generation of our in-house developed wireless IMU. Right: 3D-printed case, which contains the PCB (IMU #9, case version 11).

core, and a low-power sensor-controller core for interacting with peripheral components without requiring the main core. The wireless IMU features a single-chip IMU sensor containing both a triaxial accelerometer and gyroscope (BMI160 by Bosch), an additional high-precision accelerometer (ADXL355BEZ by Analog Devices), and a magnetic sensor (MMC3416xPJ by MEMSIC). The magnetic sensor is used for magnetic, angular rate, and gravity (MARG) sensing applications and enables distortion and gyroscope bias drift compensation. A barometric pressure sensor (BMMP388 by Bosch) is included for floor-level detection. A Serial-to-USB interface (FT230XQ) facilitates wired data transmission charging of the integrated 120mA h lithium-ion battery.

Component placement and routing are realized on a four-layer printed circuit board (PCB) with two component sides and dimensions of 34.5mm by 18mm. The populated PCB and its 3D-printed enclosure are depicted in Figure 1.

All sensors are sampled with a rate of 100 Hz, and raw sensor data is fused using an algorithm published by Madgwick for either IMU or MARG sensing applications [12]. Sensor fusion can be performed directly on the microcontroller if desired and reduces the amount of data to be transmitted significantly compared to the transmission of raw sensor data. This allows for the simultaneous operation of multiple wireless IMUs at a refresh rate of 100 Hz.

IV. CALIBRATION IMPROVEMENTS

A. Orientation Helper Object

The calibration algorithm requires at least nine different orientations to construct a well-defined optimization problem [6], [13]. In Tedaldi’s original work, the IMU is placed in $37 \leq N \leq 50$ distinct static positions, each held for 1 – 4s.

Without support, a cuboid IMU case can only be placed on six faces – or less if some are rounded, as seen in Figure 1. In [6], Tedaldi et al. show an IMU with a cable attached resting on the edge of a slab, which is a very unstable position for an IMU weighing only a few grams. The rigid cable can move the lightweight IMU by tiny amounts, preventing the algorithm from identifying static phases.

To avoid these problems in our evaluation, we use wireless transmission (see also Section IV-D) and a 3D-printed



Fig. 2: The icosahedral enclosure allows the IMU to rest in 20 different orientations. It can hold the IMU case (orange) and has a hole opposite for pushing the IMU back out.

icosahedral orientation helper object based on Tim Edwards’ OpenSCAD model [14] as shown in Figure 2. The 3D-printed regular icosahedron has a distance of 66 mm between opposing vertices. The enclosed IMU is press-fit into a slot on one of the triangular faces and can be released by pushing through a hole on the opposing face. The orientation helper object allowed us to place the IMUs to be calibrated in 20 easily reproducible orientations with at least 42 degrees rotation between them and without interfering cables. This enabled a more systematic approach to capturing calibration sequences, as explained in Section V.

B. Improved Static Phase Selection

Running the calibration algorithm [9] on our captured sequences of the low noise accelerometer revealed a weakness of the static phase detection algorithm. This issue was found in both the original C++ code [8] and the MATLAB implementation [9]. The variance of the long static phase in the beginning ς_{init} is used as a baseline for finding a suitable variance threshold for the short static segments later in the sequence. However, ς_{init} is so small due to the low noise floor of the ADXL355 that even tiny perturbations in the short sequences are above the maximum threshold of $10\varsigma_{\text{init}}$ variance. Increasing the maximum variance threshold factor from 10 to 225 solved this problem for our case but revealed another problem: v In Tedaldi’s algorithm, the best static phase threshold (an integer multiple $k \cdot \varsigma_{\text{init}}$) is determined by performing a non-linear least-squares minimization of the accelerometer cost function $L(\theta^{\text{acc}})$ for each k and selecting the k with the smallest residual. As stated in [6], this approach does not require parametrization, but favors calibration sequences in which the same orientation is repeated multiple times. Thus, the algorithm would select a k for which the long sequence at the beginning was split into multiple smaller segments of length > 1 s because the variance $\varsigma(t)$ was too close to $k\varsigma_{\text{init}}$, crossing the threshold multiple times. This problem can be fixed by rejecting segments where the acceleration vector

direction did not change relative to the previously accepted segment, minimizing the number of redundant segments.

However, the selection of the k with the minimum residual was still susceptible to the slightest perturbation of the accelerometer data. When truncating the calibration sequence even slightly, the algorithm would seemingly at random select very different values of k , which led to wildly varying numbers of segments: Fewer segments for smaller k because static phases would be broken into multiple parts, which were then rejected for being shorter than 1 s. Thus, we modified the static phase selection to always select the k which produced the largest number of usable static segments, excluding segments that were duplicates or too short. If there are multiple k , we select the one with the lowest residual.¹

C. Division by zero

The MATLAB implementation [9] contained a division by zero in the function `fromOmegaToQ` when the angular rates were $[0, 0, 0]$ in a single packet. This is a very unlikely event due to the measurement noise, but it occurred in at least one calibration sequence, so we corrected the issue.

D. Wireless Packet Loss Correction

For data transmission between IMU and host PC, we used Bluetooth 5.0 Low Energy (LE) Generic ATtribute Profile (GATT) notifications since they provide a significant improvement in data transmission speed compared to indication messages [15]. Even though the Bluetooth link layer L2CAP provides acknowledgments and retransmissions, any wireless transmission includes a risk of packet loss. Depending on the application, the loss of a single quaternion may not be critical, but the loss of raw gyroscope data in a calibration sequence will result in integration errors. Loss of accelerometer data is not critical because it is only sampled in static periods.

Thus, we implemented simple and a power-efficient erasure code (EC) for forward error correction of lost gyroscope samples in order to prevent this problem. The EC data E_i is the same size as the raw gyroscope data i (6 bytes) and consists of the XOR of the previous M raw gyroscope values $E_i = G_{i-1} \oplus G_{i-2} \oplus \dots \oplus G_{i-M}$. This allows the receiver to reconstruct arbitrary packet losses in a window of length M when followed by a sequence of M correctly received packets. Computing E_i for a new packet consists of only two XOR operations: One for adding the next G_{i-1} and one for removing the oldest element G_{i-M} from the running XOR sum. Apart from the running sum E_i , only an additional ring buffer for storing $G_i \dots G_{i-M+1}$ is required on the IMU.

V. EVALUATION

To determine the minimum required number of static positions N for a sufficiently accurate calibration, we recorded five calibration sequences with each of our four working IMUs (units #1, #2, #6 and #9) with $N \geq 37$. For each sequence, we recorded the following raw IMU data: Packet index, ADXL355

¹Our changes to the MATLAB code, including evaluation scripts, will be published under https://github.com/IMS-AS-LUH/imu_tk_matlab

acceleration, BMI160 acceleration, and BMI160 gyroscope values. The initial static phase was 40 s long, followed by static segments of ≥ 3 s. For the first 20 orientations, we placed the icosahedron on all 20 faces, with increasing numbers pointing up. After the ascending sequence, we placed the icosahedron on random faces until at least 37 poses were recorded. One calibration run of IMU #9 contained only 34 usable poses, so we discarded all results for IMU #9 for $N > 34$. For each IMU, we performed a full calibration for each captured sequence with maximum N using our improved MATLAB code, resulting in 4 times five sets of 18 calibration parameters $\theta = [\theta^{\text{acc}}, \theta^{\text{gyro}}]$, as described in [6]). For each IMU, we calculated the mean parameters θ_{mean} over the five sets, which were subsequently regarded as the ‘reference’ calibration coefficients for that IMU.

We then incrementally truncated each individual sequence, thereby reducing the effective $N = N_{\text{eff}}$. For each truncated sequence, we performed another calibration run with N_{eff} segments and compared the resulting $\theta_{N_{\text{eff}}}$ to θ_{mean} by calculating the mean absolute difference for each subset of coefficients. This allows an assessment of how the calibration quality deteriorates with decreasing N , relative to the ‘full’ calibration recommended by Tedaldi, without needing an absolute ‘gold standard’ reference of the calibration parameters.

We also captured an additional set of calibration runs without the icosahedron just by placing the IMUs on the five orthogonal faces of their case in four different horizontal attitudes. The calibration using these sequences completed successfully, but the calibration error for these runs was much larger than θ_{mean} , so we omitted the results.

VI. RESULTS

The results of the evaluation are shown in Figures 3, 4 and 5. Since the conclusion is the same for the BMI160 accelerometer, we only show the results from the calibration runs using the ADXL355 values. For the accelerometer

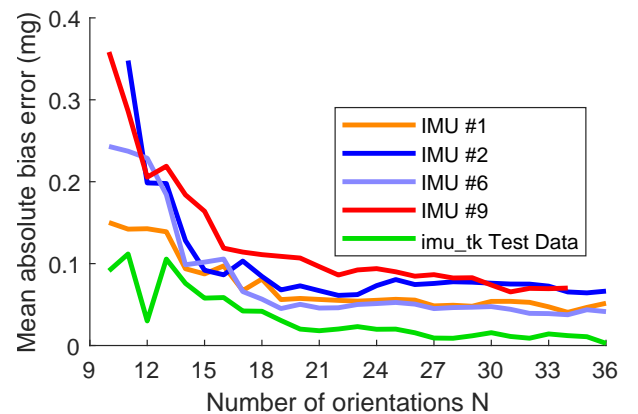
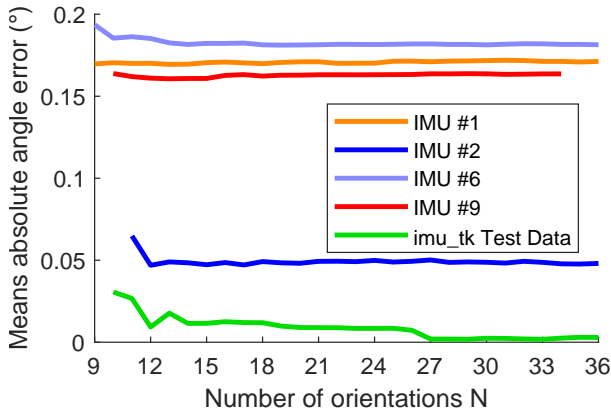
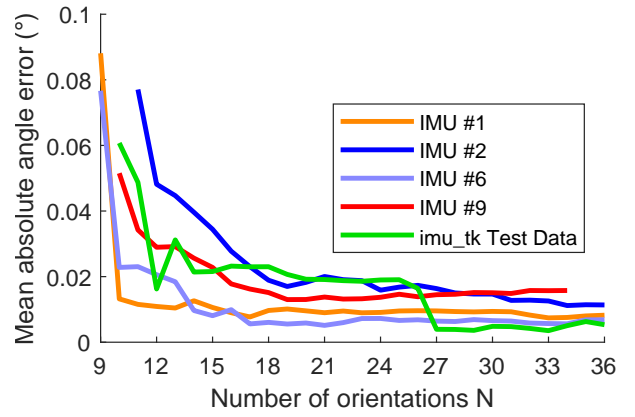


Fig. 3: Mean absolute difference of accelerometer bias estimation to θ_{mean} for varying N . Mean of 5 calibration runs for each IMU except imu_tk Test Data (only 1 calibration run). Units are $1 \text{ mg} = 9.807 \times 10^{-3} \text{ m/s}^2$

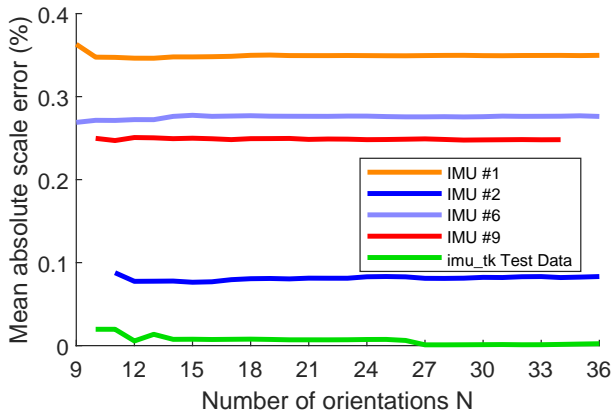


(a) Gyroscope axis misalignment estimation error over N

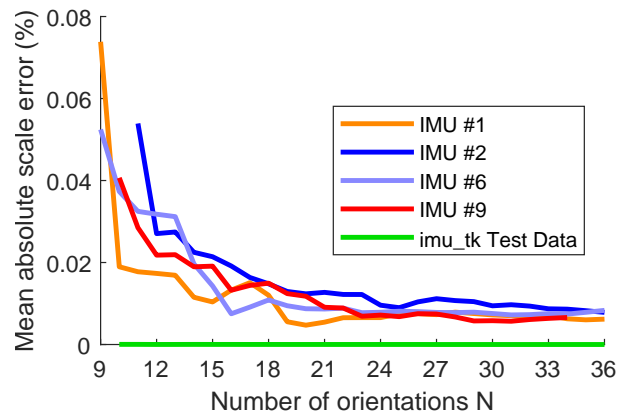


(b) Accelerometer axis misalignment estimation error over N

Fig. 4: Axis misalignment estimation errors (including non-orthogonality) for different N , when compared to θ_{mean} . Typical mean absolute misalignment angles of θ_{mean} for our IMUs were in the range between 0.4 and 0.6 degrees.



(a) Gyroscope scaling factor estimation error over N



(b) Accelerometer scaling factor estimation error over N

Fig. 5: Scaling factor estimation errors for different numbers of orientations N , when compared to θ_{mean} . Values are given in percent of the measured quantity (angular rate and acceleration respectively).

calibration, we see a slight decrease in error with larger N . However, the errors around $N = 12$ orientations are already less than 0.1% of the reference values from θ_{mean} for scaling and misalignment. For bias, the calibration error is already close to the noise floor of the ADXL355 (200 μg at 100 Hz). Thus, it is questionable whether the additional effort for triple the orientations is justified.

For the gyroscopes, the results are even clearer: As soon as enough orientations have been captured to compute a successful calibration, the accuracy barely increases at all when more positions are considered.

The results for imu_tk Test Data are unrealistically low because θ_{mean} is calculated from only one calibration run instead of five, so the calibration sequence is essentially compared to itself. Despite this, the main conclusion for N holds for this IMU as well.

VII. CONCLUSIONS AND OUTLOOK

In this paper, we presented improvements to a calibration procedure for wireless IMUs which does not require expensive

calibration equipment. The procedures presented in the initial publication by Tedaldi et al. [6] and its MATLAB implementation [9] were improved upon by correcting errors occurring in rare instances and enhancing the static phase detection when using low-noise accelerometers. A low-cost and low-accuracy 3D-printed icosahedron served as the only additional and optional calibration equipment and enabled easy positioning of the IMU in 20 distinct attitudes, although our results show that a dodecahedron would probably suffice. Evaluation of the calibration procedure showed that positioning the IMU in $N = 12$ distinct position already results in a minimal error in for the gyroscope calibration and a very small error for the accelerometer N as recommended by Tedaldi et al. [6].

In the next step, it is planned to implement the algorithm presented by Tedaldi et al. directly on the CC2652R1 SoC of our in-house wireless IMU using an approach similar to Peng et al. [11], but without sacrificing gyroscope and axis misalignment calibration. For this purpose, the algorithm's structure will need to be completely remodeled, optimizing for computational complexity, program size, and RAM usage.

REFERENCES

- [1] X. Hu and X. Qu, "Pre-impact fall detection," *BioMedical Engineering OnLine*, vol. 15, no. 1, Jun. 2016. [Online]. Available: <https://doi.org/10.1186/s12938-016-0194-x>
- [2] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, "Reviews on various inertial measurement unit (imu) sensor applications," *International Journal of Signal Processing Systems*, vol. 1, no. 2, pp. 256–262, 2013.
- [3] Y. Ganesan, S. Gobee, and V. Durairajah, "Development of an upper limb exoskeleton for rehabilitation with feedback from EMG and IMU sensor," *Procedia Computer Science*, vol. 76, pp. 53–59, 2015. [Online]. Available: <https://doi.org/10.1016/j.procs.2015.12.275>
- [4] C. Malleson, A. Gilbert, M. Trumble, J. Collomosse, A. Hilton, and M. Volino, "Real-time full-body motion capture from video and imus," in *2017 International Conference on 3D Vision (3DV)*, 2017, pp. 449–457.
- [5] S. Zihajehzadeh and E. J. Park, "A novel biomechanical model-aided imu/ubw fusion for magnetometer-free lower body motion capture," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 6, pp. 927–938, 2017.
- [6] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for imu calibration without external equipments," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3042–3049, 2014.
- [7] J. Botero-Valencia, D. Marquez-Viloria, L. Castano-Londono, and L. Morantes-Guzmán, "A low-cost platform based on a robotic arm for parameters estimation of inertial measurement units," *Measurement*, vol. 110, pp. 257–262, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0263224117304360>
- [8] A. Pretto, "imu_tk c++ implementation," 2014. [Online]. Available: https://bitbucket.org/alberto_pretto/imu_tk/
- [9] J. Huai, "imu_tk_matlab implementation," 2017. [Online]. Available: https://github.com/JzHuai0108/imu_tk_matlab
- [10] C. Ren, Q. Liu, and T. Fu, "A novel self-calibration method for mimu," *IEEE Sensors Journal*, vol. 15, no. 10, pp. 5416–5422, 2015.
- [11] C.-C. Peng, J.-J. Huang, and H.-Y. Lee, "Design of an embedded icosahedron mechatronics for robust iterative imu calibration," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 3, pp. 1467–1477, 2022.
- [12] S. Madgwick *et al.*, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," *Report x-io and University of Bristol (UK)*, vol. 25, pp. 113–118, 2010.
- [13] Z. F. 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*, vol. 18, no. 7, pp. 1897–1907, may 2007. [Online]. Available: <https://doi.org/10.1088/0957-0233/18/7/016>
- [14] T. Edwards, "Openscad polyhedral dice," 2015. [Online]. Available: <https://www.thingiverse.com/thing:1043661>
- [15] *Generic Attribute Profile (GATT)*, Bluetooth SIG, 1 2022, rev. GATT.TS.p21.