Real-Time 3D Magnetometer Calibration for Embedded Systems Based on Ellipsoid Fitting

Salvatore Tedesco *, Javier Torres-Sanchez, Brendan O'Flynn
Wireless Sensor Networks Group, Micro & Nano Systems Centre, Tyndall National Institute
University College Cork, Cork, Ireland
Email: salvatore.tedesco@tyndall.ie

Abstract— **Magnetometers are massively diffused in a variety of instruments for different applications. However, these sensors are affected by non-idealities, especially bias and hard/soft-iron interference. Even though a number of calibration approaches could be adopted to compensate for magnetic interferences, their implementation on resources-constrained platforms is still problematic due to the highly complex, computationally intensive mathematical operations involved. The present work demonstrates the possibility to develop a fast and efficient 3D magnetometer calibration for real-time embedded systems with limited system resources. A number of techniques and approaches are discussed to mitigate the computational burden. Results confirm that this is achievable by preserving the same level of accuracy reached with more computationally intensive approaches. Results are also promising regarding the adoption of the discussed method on low-power real-life systems in several applications.**

Keywords—3D Magnetometer, Real-Time Embedded Systems, On-the-Fly Calibration, Ellipsoid Fitting.

I. INTRODUCTION

Nowadays, magnetometers are massively diffused in a variety of instruments for different applications, e.g., automotive, space and military research, navigation, etc [1]. Their use in consumer products has further increased the research in the area. However, low-cost magnetometers are typically influenced by nearby magnetic perturbations in everyday environments. Thus, on-the-fly compensation is essential to achieve accurate performance. This requires that error parameters (e.g., bias, scale factors) are accurately estimated from the data, which can be challenging on devices with limited system resources, in terms of memory and speed.

Traditional methods [2-3] may rely on external reference information, i.e., GPS, inertial sensors (accelerometers and gyroscopes), etc. but those are not always available and, thus, these methods are insufficient and costly.

Other approaches consider a complete error modeling, which is computationally intensive and not suitable for battery-powered portable devices. A well-known example is the ellipsoid fitting algorithm [4-5]. As already proved in literature [6], ellipsoid fitting algorithms have better performance compared to other low-cost approaches, such as multi-attitude or neural networks.

Given that the locus of the magnetometer data, when only the Earth magnetic field is present, forms a sphere which is transformed into an ellipsoid in the presence of magnetic interference, the method defines the scaling and bias components to reverse this transformation. The algorithm works both indoor/outdoor and only requires to rotate the sensor in the 3D space without using further equipment. This method can be solved via linear least-square or iterative approaches, which need large amount of computation [7-10].

Moreover, even with computationally efficient ellipsoid fitting approaches, redundant measurements of the magnetic field in a wide range of orientations need to be stored to achieve the desired precision, which implies significant resources for data collection and storage. Thus, the implementation of fast on-the-fly magnetometer compensation methods is still under investigation. Even though the work presented in [11] shows the possibility to carry out magnetometers calibration on embedded platforms, such as a STM32 microcontroller, no indications are provided on how the authors overcome numerical problems related to data storage, matrix inversion, and eigensystems resolution.

The present work describes a computationally efficient implementation of the ellipsoid fitting method for real-time limited-resource embedded systems. A detailed description of the numerical problems involved in the ellipsoid fitting method will be provided, along with suitable solutions for low-power resource-constrained devices. The ellipsoid fitting modelling will be discussed in Section II, while the techniques considered to optimize and improve the required computation are analyzed in Section III. Final results are then presented and discussed in Section IV.

II. ELLIPSOID-FITTING MODELLING

For a 3D magnetometer, the error is mainly due to bias, scale factor, non-orthogonalities, and deviations influenced by local interferences. The physical magnetic field h_t, and the sensor data h_s, in presence of error, are connected as shown in Eq. (1)

$$h_s = K_e h_t + B_e \quad (1)$$

where K_e takes into account the sensitivity of the sensor, non-orthogonality and soft-iron errors, B_e represents the bias and hard-iron errors. As h_t is the desired parameter, the equation is inverted as shown in Eq. (2),

* Corresponding author

$$h_t = K_c(h_s + B_c) \quad (2)$$

where $K_c = K_e^{-1}$ and $B_c = -B_e$.

Within a fixed area, the magnitude of the physical magnetic field remains constant, and when the sensor rotates, the locus of the measured magnetic field represents a sphere where

$$\|h_t\|^2 = H^2 \quad (3)$$

i.e., H is the intensity of the local magnetic field. Thus:

$$F(\alpha, h_s) = \varepsilon^T\alpha = h_s^T A h_s - 2b^T A h_s + b^T A b - H^2 = 0 \quad (4)$$

where $A = K_c^T K_c$ and $b = -B_c$. Eq. (4) then represents an ellipsoid equation on vector $h_s = [h_{s,x}, h_{s,y}, h_{s,z}]$, with $\alpha = [a_1, a_2, \dots, a_{10}]^T$ being the ellipsoid parameters, $\varepsilon = [h_{s,x}^2, h_{s,x}h_{s,y}, h_{s,y}^2, h_{s,x}h_{s,z}, h_{s,y}h_{s,z}, h_{s,z}^2, h_{s,x}, h_{s,y}, h_{s,z}, 1]^T$ and

$$A = \begin{bmatrix} a_1 & a_2/2 & a_4/2 \\ a_2/2 & a_3 & a_5/2 \\ a_4/2 & a_5/2 & a_6 \end{bmatrix} \quad (5)$$

$$b = -\frac{1}{2}[BA^{-1}]^T \quad (6)$$

$$B = [a_7, a_8, a_9] \quad (7)$$

$$a_{10} = b^T A b - H^2 \quad (8)$$

The objective of the magnetometer calibration is to derive the coefficient matrix A and spherical center coordinates b based on a set of N magnetic measurements, finding an ideal ellipsoid which minimizes the sum of squares of the algebraic distances $F(\alpha, h_s)$ from the measured data to the ellipsoid, e.g., $\min_{\alpha} \alpha^T D^T D \alpha$, where

$$D = \begin{bmatrix} h_{s,x1}^2 & h_{s,x1}h_{s,y1} & h_{s,y1}^2 & h_{s,x1}h_{s,z1} & h_{s,y1}h_{s,z1} & h_{s,z1}^2 & h_{s,x1} & h_{s,y1} & h_{s,z1} & 1 \\ \vdots & & & & \vdots & & & & & \vdots \\ h_{s,xN}^2 & & & & \dots & & & & & 1 \end{bmatrix} \quad (9)$$

This problem can be solved directly by the standard least squares approach, but the result of such fitting is a general conic. To be an ellipse the solution of this problem is subject to a specific constraint which, without losing generality, can be expressed as $\alpha^T C \alpha = 1$, with C defined as

$$C = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (10)$$

The minimization problem is solved by applying the Lagrange multipliers, where the following conditions for α are obtained:

$$\begin{cases} S\alpha = \lambda C\alpha \\ \alpha^T C \alpha = 1 \end{cases} \quad (11)$$

where S is the 10x10 scatter matrix given by $D^T D$. It can be proved that only one of the generalized eigenvalues in Eq. (11) is positive, and its eigenvector α is the best ellipsoid fitting parameter.

An effective and robust algorithm for an ellipse-specific fitting of data points is obtained through a subdivision of the matrices involved, so as to divide Eq. (11) in two subsystems of reduced order. Indeed, S and C have special structures which allow a simplification of the problem. Scatter matrix S and constraint matrix C can be split as follows:

$$S = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}, C = \begin{bmatrix} C_1 & 0_{3x7} \\ 0_{7x3} & 0_{7x7} \end{bmatrix}, C_1 = \begin{bmatrix} 0 & 0 & 2 \\ 0 & -1 & 0 \\ 2 & 0 & 0 \end{bmatrix} \quad (12)$$

Where S_{11} is 3x3 and S_{22} is 7x7, both symmetrical, and $S_{12} = S_{21}^T$. Finally, also vector α can be split into $\alpha = [\alpha_1, \alpha_2]^T$, with $\alpha_1 = [a_1, a_2, a_3]^T$ and $\alpha_2 = [a_4, \dots, a_{10}]^T$.

Due to Eq. (12), $\alpha^T C \alpha = 1$ becomes $\alpha_1^T C_1 \alpha_1 = 1$, while $S\alpha = \lambda C\alpha$ is shaped as

$$\begin{cases} S_{11}\alpha_1 + S_{12}\alpha_2 = \lambda C_1 \alpha_1 \\ S_{12}^T \alpha_1 + S_{22}\alpha_2 = 0 \end{cases} \quad (13)$$

Matrix S_{22} is singular if all data lie on a line, otherwise, it is regular and invertible, thus $\alpha_2 = -S_{22}^{-1} S_{12}^T \alpha_1$. After some manipulations, the first part of Eq. (13) yields to $M\alpha_1 = \lambda\alpha_1$, where $M = C_1^{-1}(S_{11} - S_{12}S_{22}^{-1}S_{12}^T)$ is a 3x3 reduced scatter matrix, not symmetric, positive definite. Eq. (13) then becomes

$$\begin{cases} M\alpha_1 = \lambda\alpha_1 \\ \alpha_1^T C_1 \alpha_1 = 1 \end{cases} \quad (14)$$

Hence, the algorithm for performing the magnetometer compensation firstly considers the collection of N sample measurements of the surrounding magnetic field h_s, from which matrices D and S are defined. Then, matrix M is obtained, and the reduced eigensystem in Eq. (14) is solved finding the eigenvector α_1 associated to the only positive eigenvalue. Next, α_2 can be defined and, thereby, the obtained eigenvector α is used to solve matrices A, B, and b as in Eq. (5-8). The compensation coefficients K_c (via Cholesky decomposition) and B_c are finally obtained, and then the error-free vector h_t.

III. Real-Time Implementation

For a real-time system, the most computationally expensive steps of the algorithm in Section II are related to the storage and analysis of the magnetic data (e.g., matrices D and S), the inversion of S_{22}, and the resolution of the eigensystem in Eq. (14). Those points will be discussed separately in this section.

A. Data Storage

The main aspect in data storage is related to the definition of the scatter matrix S, which is expensive to implement as it involves the multiplication of matrix D with its transpose with size Nx10 (N being the number of measurements taken by the magnetometer, which are in the order of hundreds), and the storage of those matrices on-board. The proposed solution is based on the online update of S for every new obtained magnetic sample.

Given the matrices X, of size n×m, and Y, of size m×p, their product XY_m has size n×p. If an additional column is added to X (and consequently a new row is added to Y), their new product XY_{m+1} can be obtained from XY_m by adding a second matrix Z constituted by all the possible combinations of the elements which are part of the added column m+1, where

$$Z = \begin{bmatrix} X_{1,m+1}Y_{m+1,1} & \dots & X_{1,m+1}Y_{m+1,p} \\ \vdots & \vdots & \vdots \\ X_{n,m+1}Y_{m+1,1} & \dots & X_{n,m+1}Y_{m+1,p} \end{bmatrix} \quad (15)$$

Since, in the studied scenario, Y is the transpose of X, n = p = 10, and m is the continuously increasing number of measurements, then Z is symmetric requiring only 55 elements out of 100 to be calculated.

Thus, given the first two magnetic measurements, it is possible to define the matrix D (Eq. (9) with N = 2) and the related scatter matrix S. When a new magnetic sample is obtained, those latest data are arranged in a row vector defined as ε which is multiplied for its transpose to obtain a 10x10 matrix representing Z. This matrix is finally added to the previously calculated S so as to obtain an updated scatter matrix. The process is repeated every time a new magnetic sample is gathered.

It is worth noting that, in this process, only 110 FLOPs (Floating point operations per second [12]) are requested (55 additions and 55 scalar multiplications), and this value is constant for every data sample collected, while 300 FLOPs are needed for estimating the initial scatter matrix when N=2.

The brute-force approach of calculating off-line the multiplication of two matrices of size 10xN and Nx10 requires 200N – 100 FLOPs, which can be reduced to 110N – 55 if considering the assumption of symmetry. Instead, this method requires overall 300 + 110(N-2) = 110N + 80 FLOPs. With an addition of only 135 FLOPs per magnetic sample, this method makes the calculation online possible eliminating the need to store neither large numbers of magnetic samples on memory nor the updated scatter matrix, as it can be continuously overwritten.

B. Matrix Inversion

Matrix M, defined in Section II, requires the inversion of the 7x7 matrix S_{22}, which, because of the number of mathematical operations, can be challenging on constrained systems. The Gauss-Jordan method results in 364 FLOPs, relying on integrated standard libraries, which however should be avoided to guarantee faster boot time and smaller memory requirements. Moreover, the inverted matrix is then multiplied with 7x3 matrix S_{12}^T, for additional 273 FLOPs.

Thus, based on the fact that solving a linear system is faster than inverting a matrix, it is advisable to directly solve $S_{22}^{-1}S_{12}^T = G$ expressed as $S_{22}G = S_{12}^T$.

Assuming the Cholesky decomposition of S_{22}, e.g., $S_{22} = LL^T$, with L a lower triangular matrix, after some manipulations it is obtained that

$$L^T G = y, \text{ and } Ly = S_{12}^T \quad (16\ a\text{-}b)$$

While Eq. (16b) can be firstly solved with forward substitution, the resulting 7x3 matrix y can be used to solve Eq. (16a) via back substitution finally obtaining G. The forward/back substitutions require overall 294 FLOPs, while the Cholesky decomposition for a 7x7 matrix needs 140 FLOPs.

Hence, S_{22} matrix inversion followed by S_{12}^T matrix multiplication, using standard techniques, would result in 637 FLOPs, whereas in the proposed method inversion and multiplication are calculated implicitly without any time penalty for 434 FLOPs.

C. Eigensystem Resolution

In numerical analysis, one of the most important problems is designing efficient and stable algorithms for finding the eigenvalues and eigenvectors of a matrix. Unfortunately, there is no simple algorithm to directly calculate eigenvalues for general matrices, and that is the reason why iterative algorithms are typically used to solve the eigenvalue problem by producing sequences that converge to the eigenvalues. The QR algorithm is a very powerful algorithm to stably compute the eigenvalues and (if needed) the corresponding eigenvectors, and still today it is by far the most popular approach for solving dense non-symmetric eigenvalue problems.

Matrix M, as defined in Section II, is dense and non-symmetric, and the QR algorithm [13-14] is the most popular approach for solving this type of problem. It can be proved that a standard QR step with a nxn Hessenberg matrix gains an order of magnitude in terms of operations in relation to a QR step with a full matrix [15]. An upper/lower Hessenberg matrix is a square matrix where values below/above the subdiagonal are zero.

A typical method used to convert a general matrix into a Hessenberg matrix with the same eigenvalues is called Householder reduction. In the studied case, M has dimension 3, and thus it is simple to define a closed-form formula for the Householder transformation [13]. The reduction to Hessenberg form costs $10/3\,n^3$ FLOPs, which is equivalent to 90 operations in the considered case.

Additionally, the convergence of the Hessenberg QR algorithm can be improved by introducing (spectral) shifts into the algorithm. In this scenario, it has been considered the Francis' double shifts QR algorithm with deflation steps (incorporated by declaring as zero the last lower off-diagonal element when sufficiently small, so that the algorithm proceeds with a smaller matrix). This method [14, Chapt. 7, pp. 358] is characterized by an improved convergence. The outcome is a triangular matrix whose diagonal values represent the requested eigenvalues.

Generally, it is not simple to define the complexity of this method, but assuming that two steps are requested per each eigenvalues, and counting also the operations requested for the Householder reduction, it can be estimated that the complexity for this scenario (with non-symmetric matrix and without calculating the eigenvectors) is equal to $10n^3$ (or $25n^3$ if also eigenvector calculation is considered). This consideration is significant in highlighting the appropriate ellipsoid fitting modelling discussed in Section II which involves the definition of M as a 3x3 matrix, differently from what shown

in [11] where the adoption of a different matrix C involves a 6x6 matrix M. This indicates that the latter case is 8 times more computationally demanding.

In this algorithm, eigenvectors are not needed for all eigenvalues but only for the unique positive one. Once this eigenvalue is detected, it is then simple to define the related eigenvector α_1 by solving Eq. (14) by substitutions.

IV. Results and Discussion

This section compares the discussed algorithm, implemented as described in Section III, and a standard ellipsoid-fitting algorithm [16], in terms of execution time and accuracy. The solution proposed in [16] is efficient and based on the Singular Value Decomposition (SVD), which is a numerically safe technique without any problem of convergence, but also with a high computational burden especially if applied on large matrices. Extensive test on Matlab and embedded platforms have been carried out.

To compare the two algorithms, 3D magnetic data have been collected several times with a magnetometer at 80 Hz by moving the sensor in all directions in an environment surrounded with ferromagnetic materials (e.g., typical office environment). The two methods have been implemented in Matlab, showing similar results in terms of accuracy. Indeed, the maximum relative difference for the diagonal elements of the matrix K_c was 1.38% and 1.42% for the vector B_c, indicating very good correlation. The off-diagonal elements in K_c, instead, are averagely 250 times lower compared to the on-diagonal elements, which can then be deflated.

The analysis has been repeated 1000 times running on a Intel® Core™ i5-4570 CPU 3.20 GHz with 8 GB RAM and 64-bit OS, and the average execution times for a single iteration were 193.3 s for [16] (st. dev. 9.7 s), and 0.56 (st. dev. 0.06 s) for the described algorithm, e.g., 350 times lower.

Fig. 1-2-3 shows an example of the calibration outcome, by including the raw data (in blue) and the calibrated ones (in red). It can be noted how the blue points show an ellipsoid while the red figure is equivalent to a unit spheroid.

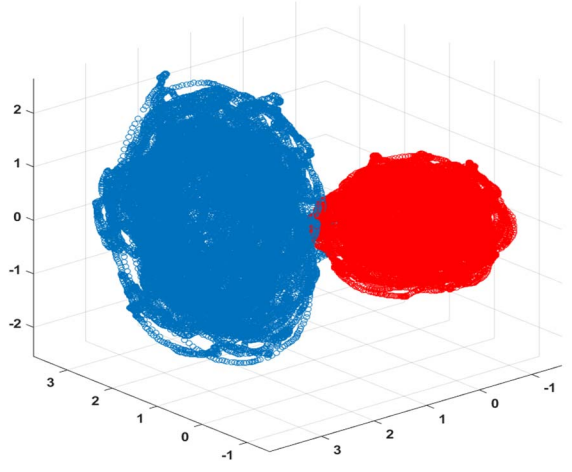

Fig. 1. Raw (blue points) and calibrated (red points) data. Calibrated data are normalized to a unit spheroid, and the raw data are normalized for visualization purposes

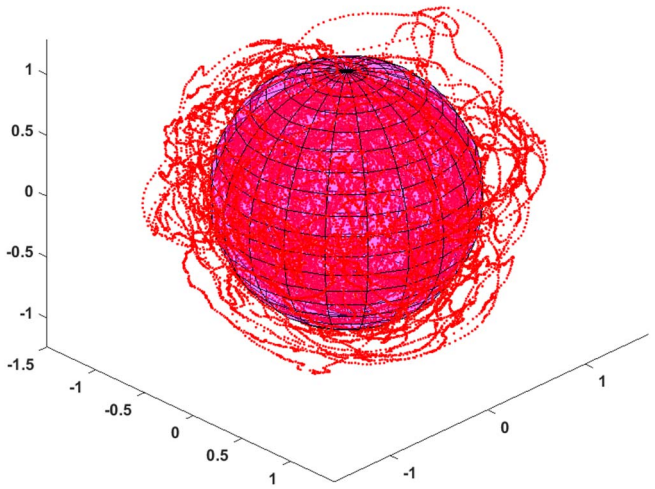

Fig. 2. Calibrated data along with the fitting unit spheroid

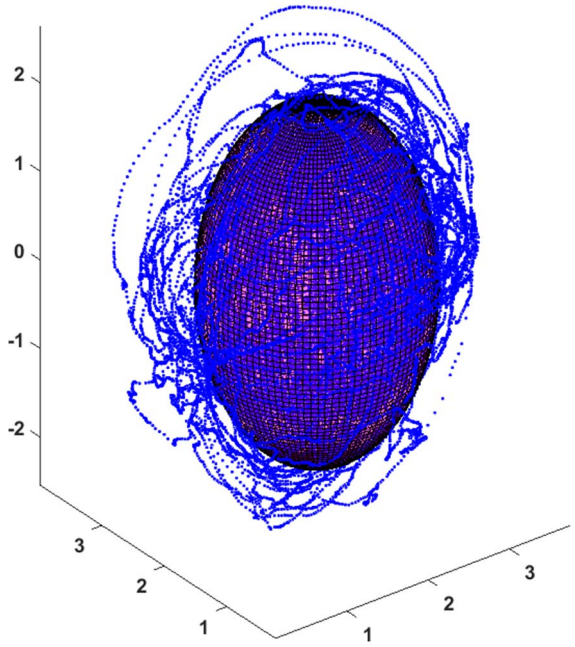

Fig. 3. Raw data (normalized for visualization purpose) along with the fitting ellipsoid

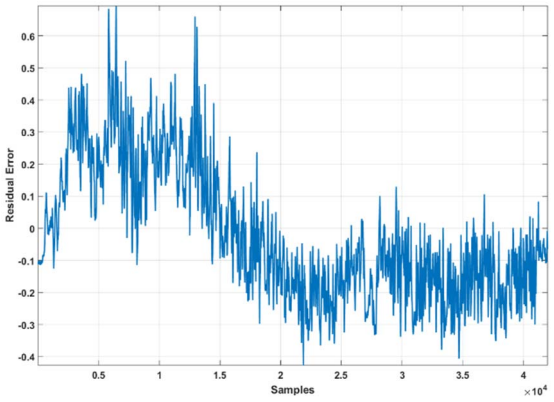

Fig. 4. Residual error after calibration

Moreover, in order to quantify the quality of the calibration process, the residual error has been calculated for all the methods. The residual error represents the distance between each of the calibrated data and the closest point lying on the surface of the spheroid. The average residual error (Fig. 4) is equal to -0.0208 (st. dev. 0.2027) for the described algorithm, and -0.0855 (st. dev. 0.1736) for [16], again showing good correlation.

Furthermore, to confirm that the approaches discussed in Section III make the algorithm more efficient on a platform with limited system resources, various tests have been performed.

Firstly, the scatter matrix S has been obtained in Matlab from gathered magnetics data using both off-line approach and the method illustrated in III.A. taking one data sample at the time. The resulting scatter matrices with the two methods have a maximum difference of only 2.91e-13, proving identical performance even though the method in III.A is more suitable for real-time systems.

Similarly, matrix G calculated with a standard technique (Gauss-Jordan inversion followed by a matrix multiplication) has a maximum difference of 2.84e-13 when compared with the same variable calculated with the more efficient method shown in III.B (e.g. Cholesky decomposition and back/forward substitutions).

Finally, the Hessenberg form matrix resulting from the proposed approach in III.C. is significantly similar to the one obtained with standard functions in Matlab, with a maximum error of 1.49e-08. As per the QR algorithm, the maximum difference between the eigenvalues estimated with standard functions and the one obtained with the method described above is 5.37e-04, thus, proving the equivalence between the two methods.

The algorithm has been also implemented on a low-power Atmel 32-bit AVR UC3 microcontroller with low CPU clock frequency (48 MHz) and single precision HW floating point unit used on a glove-like system [17-18] with 16 3D magnetometers. The sensors adopted were the MPU-9250 [19]. The system collects data, runs the algorithm on-board in real-time, and sends the calibration results and the magnetic data to a PC, for further off-line analysis. Results comparing matrices K_c and B_c calculated on-board and via Matlab are shown in Table I depicting identical performance. The execution time of the real-time soft-iron embedded algorithm for every new magnetometer sample has been measured with an oscilloscope, resulting in 525 µs. When working simultaneously with all 16 sensors the overall latency is then still limited to less than 10 ms, suitable for several applications.

Finally, the 3D sensor orientation has been estimated while rotating the glove device over the vertical axis on a non-magnetic swivel chair (so as to modify only the heading/yaw angle). The sensor was strapped and aligned with the chair which was rotated a number of times. The algorithm discussed in [20] is used for the estimation of the orientation. The orientation is calculated on-board simultaneously with and without magnetic real-time compensation parameters taken into account. The heading error is shown in Fig. 5 and it proves the reliability of the implemented magnetic compensation on-board algorithm. The heading error during the rotations is included between -5.5 and 7 deg, with average value equal to 1.32 deg.

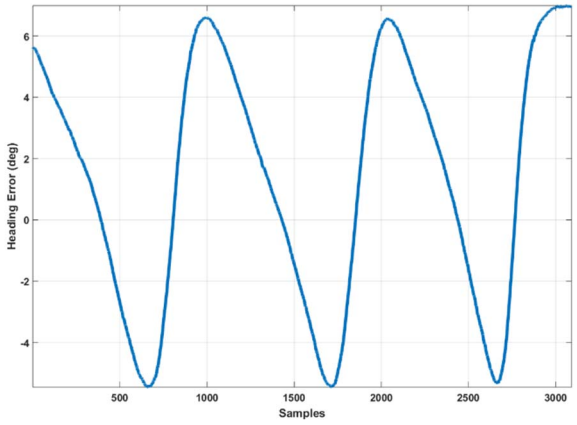

Fig. 5. Heading error w/wo magnetic calibration

These results show that the developed real-time compensation algorithm is in good correlation when compared to post-processing algorithms and is ideal for low-power embedded systems with constrained system resources. Finally, the studied method is reliable, with no problem of numerical stability and fast. Extensive Matlab analyses have been carried out to address the definition of computationally effective least squares-based ellipse-specific methods, with special consideration to modelling numerically stable routines for eigensystems resolution, matrix inversion and data collection. The algorithm, implemented on a 32-bit microcontroller, proved the feasibility of the method showing the same accuracy achieved with more computationally intensive approaches.

TABLE I. MATLAB VS. EMBEDDED RESULTS

Param.	Matlab / Real-Time	Param.	Matlab / Real-Time	Param.	Matlab / Real-Time
$K_c(1,1)$	0.0268 / 0.0268	$K_c(2,2)$	0.0282 / 0.0282	$K_c(3,3)$	0.0255 / 0.0255
$K_c(1,2)$	0 / 0	$K_c(2,3)$	0 / 0	$B_c(1)$	6.8151 / 6.7594
$K_c(1,3)$	0 / 0	$K_c(3,1)$	-0.0023 / -0.0023	$B_c(2)$	-2.5962 / -2.5751
$K_c(2,1)$	0.0002 / 0.0002	$K_c(3,2)$	0.0005 / 0.0005	$B_c(3)$	-6.8767 / -6.8218

V. CONCLUSION

The implementation of fast on-the-fly magnetometer compensation methods is still under investigation. The present paper described a computationally efficient implementation of the ellipsoid fitting method for real-time limited-resource embedded systems. A detailed description of the numerical problems involved in the ellipsoid fitting method are provided, along with a number of techniques and approaches discussed to mitigate the computational burden on low-power resource-constrained devices. Extensive test on Matlab and embedded platforms have been carried out showing significant similarities. In conclusions, the present work demonstrated the

possibility to develop a fast and stable magnetometer calibration for real-time systems with constrained system resources by preserving the same level of accuracy achieved with more computationally intensive approaches. Results confirm those conclusions and are promising regarding the application of the discussed algorithm on real-world systems.

ACKNOWLEDGMENTS

This publication has emanated from research supported by a research grant from Science Foundation Ireland (SFI) under grant SFI/12/RC/2289, the European Regional Development Fund under grant number 13/RC/2077-CONNECT, and the Enterprise Ireland funded project VR-GLOVE under grant number CF-2015-0031-P.

REFERENCES

[1] P. Ripka, "Magnetic sensors and magnetometers," Artech House, 2001.

[2] M. Kok, et al., "Calibration of a magnetometer in combination with inertial sensors," Int. Conf. FUSION, Singapore, July 2012, pp. 787-793.

[3] K. Han, et al., "Extended Kalman filter-based gyroscope-aided magnetometer calibration for consumer electronic devices," IEEE Sensors Journal, 17, 1, pp. 63-71, 2017.

[4] Y.X. Liu, et al., "Novel calibration algorithm for a three-axis strapdown magnetometer," Sensors, 14, pp. 8485-8504, 2014.

[5] R. Halir, et al., "Numerically stable direct least squares fitting of ellipses," Proc. Int. Conf. Cent. Eur. Comput. Gr., 1, pp. 125–132, 1998.

[6] Z. Xiaojuan, et al., "Comparison of three kinds of compensation algorithms based on magnetic sensors, " IEEE Int. Conf. on Electronic Measurement and instruments, 2013.

[7] S. Zhongguo, et al., "A calibration method of three-axis magnetometer with noise suppression," IEEE Transactions Magnetics, 50, 11, 2014.

[8] L. Ye, S.W. Su, "Experimental design for the calibration of tri-axial magnetometers," Int. Conf. Sensing Technology, Auckland, Dec. 2015, pp. 711-715.

[9] H. Jin, "A geometric method for calibrating three-axis magnetometers," Int. Conf. Signal Processing, Communications and Computing (ICSPCC), Xiamen, Oct. 2017.

[10] X. Cui, et al. "Three-axis magnetometer calibration based on optimal ellipsoid fitting under constraint condition for pedestrian positioning system using foot-mounted inertial sensor/magnetometer," IEEE/ION Position, Location and Navigation Symposium (PLANS), Monterey, US, Apr 2018, pp. 166-174.

[11] X. Lou, et al., "Realization of ellipsoid fitting calibration for three-axis magnetic sensor based on STM32 embedded system," Int. Conf. Human Centered Computing (HCC), pp. 717-726, 2014.

[12] https://mediatum.ub.tum.de/doc/625604/625604

[13] R.L. Burden, et al., "Numerical analysis," Brooks/Cole Pub. Co, 1997.

[14] G.H. Golub, et al., "Matrix computations," John Hopkins U Press, 1996.

[15] http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf

[16] S. Bonnet, et al., "Calibration methods for inertial and magnetic sensors," Sens Actuators A: Phys., 156, 2, pp. 302–311, 2009.

[17] B. O'Flynn, et al., "Novel smart glove technology as a biomechanical monitoring tool," Sens Trans Journal, 193, 10, pp. 23-32, Oct 2015.

[18] J. Torres-Sanchez, et al., "A 3D hand motion capture device with haptic feedback for virtual reality applications," IEEE Games, Entertainment & Media Conf. (GEM), Galway, Ireland, Aug 2018.

[19] https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

[20] S.O. Madgwick, et al., "Estimation of IMU and MARG orientation using a gradient descent algorithm," IEEE Int Conf Rehabil Robot, Zurich, July 2011.