sensors

MDPI

*Review*

# Magnetic Field Sensors' Calibration: Algorithms' Overview and Comparison

**Konstantinos Papafotis *, Dimitris Nikitas and Paul P. Sotiriadis ***

Department of Electrical and Computer Engineering, National Technical University of Athens, 157 80 Athens, Greece; el15006@central.ntua.gr
* Correspondence: kpapafotis@mail.ntua.gr (K.P.); pps@mail.ntua.gr (P.P.S.)

**Abstract:** The calibration of three-axis magnetic field sensors is reviewed. Seven representative algorithms for in-situ calibration of magnetic field sensors without requiring any special piece of equipment are reviewed. The algorithms are presented in a user friendly, directly applicable step-by-step form, and are compared in terms of accuracy, computational efficiency and robustness using both real sensors' data and artificial data with known sensor's measurement distortion.

**Keywords:** magnetic sensor; calibration; algorithms; review; comparison



**Citation:** Papafotis, K.; Nikitas, D.; Sotiriadis, P.P. Magnetic Field Sensors' Calibration: Algorithms' Overview and Comparison. *Sensors* **2021**, *21*, 5288. https://doi.org/10.3390/s21165288

Academic Editor: Nicolò Marconato

Received: 7 June 2021
Accepted: 22 July 2021
Published: 5 August 2021



## 1. Introduction

Magnetic field sensors (magnetometers) are nowadays widely used in a plethora of commercial, industrial, marine, aerospace and military applications. Their applications include but not limited to navigation and attitude estimation, geophysical surveys, archaeology, entertainment devices, consumer electronics and others.

In most applications, sensor's calibration is essential in order to achieve the desirable accuracy level. The purpose of magnetic field sensors' calibration is a twofold. First, as in the case of every measurement unit, calibration ensures that the measurement of the standalone sensor corresponds to the actual value of the magnetic field. To do so, calibration must compensate for all static (manufacturing imperfections etc.) and active (temperature, humidity, etc.) phenomena effecting the accuracy of the sensor's measurement. In addition, when a magnetic sensor is embedded in a larger system, other components of the system may cause disturbances (both static and active ones) to the local magnetic field. Static disturbances are usually caused by magnetic and ferromagnetic materials in the vicinity of the sensor; called hard-iron distortion and soft-iron distortion respectively (more information are given in Section 2). Mechanical or electronic structures embedded in the system, such as motors and coils could also actively distort the local magnetic field and cause significant measurement error.

This review paper focuses on algorithms correcting the dominant linear time-invariant (static) measurement errors, requiring no special piece of equipment for their application. Such algorithms are most commonly used for in-situ calibration of magnetic field sensors which are usually in chip form and embedded in larger systems. The paper presents seven representative calibration algorithms for three-axis magnetometers and compares them in terms of accuracy, robustness, computational efficiency and ease of deployment. The seven algorithms are briefly presented, to introduce all required mathematical expressions, and are summarized in an easy-to-develop, step-by-step form. For the details of the algorithms, the reader is referred to the original works.

The selection of the particular algorithms was done based on their popularity and on our attempt to present as many different calibration approaches as possible. The TWOSTEP [1] algorithm is one of the first algorithms that addressed the full calibration problem (and probably the most popular one). At a later time, Elkaim and Vasconcelos [2] proposed a geometric approach of TWOSTEP which is also very popular. At the same time, Dorveaux et al. [3]

offered a non-linear formulation of the problem and they treated it in an innovative, strictly iterative way. In addition, Wu and Shi [4] suggested the most complete formulation of the calibration problem as an optimal maximum likelihood estimation one. The TWOSTEP algorithm, as well as the algorithms proposed by Vasconcelos et al. and Wu et al., consist of a first step deriving an initial solution, and, a second step for improving it. On the other hand, Papafotis and Sotiriadis [5] recommended an iterative approach based on a twofold minimization, which was shown to be extremely effective. Furthermore, a real-time approach by Crassidis et al. [6] using the popular Kalman Filter is discussed. Finally, to represent the recent trends towards Machine Learning, an Artificial Intelligence (AI) method applying Particle Swarm Optimization on the estimation problem is explored [7].

Please note that this review focuses on works for in-situ calibration of three-axis magnetic field sensor without using any special piece of equipment or any other additional sensor. Thus, several interesting works dealing with magnetometer's calibration, in combination with inertial sensors, [8–12] are not included in this work.

The rest of the paper is organized as follows. First, a standard error model for three-axis magnetic field sensors is presented in Section 2. In Sections 3–9, seven representative algorithms are discussed in chronological order of publication. In Section 10, a method for generating artificial data is proposed and algorithms are evaluated via extensive Monte Carlo simulation to identify their performance. In addition, the algorithms are evaluated using several real sensor's measurements in order to evaluate their performance under real-world conditions. Finally, Section 11 summarizes our findings and provides brief comments for each algorithm. The notation used along the paper is presented in Table 1.

**Table 1.** Notation.

| | |
|---|---|
| $\|.\|$ | Euclidean Norm |
| $\|.\|_F$ | Frobenius Norm |
| vec $(\cdot)$ | Vectorization of Matrix |
| diag$(\cdot)$ | Diagonal Matrix |
| chol $(\cdot)$ | Cholesky Factorization |
| $I_{n\times n}$ | $n \times n$ Identity Matrix |
| $0_{m\times 1}$ | $m \times 1$ Zero Vector |
| $\mathcal{N}$ | Normal Distribution |
| $\mathcal{U}$ | Uniform Distribution |
| $\nabla$ | Gradient Vector |
| $\nabla^2$ | Hessian Matrix |
| $\otimes$ | Kronecker Product |
| $O(3)$ | Orthogonal Group of dimension 3 |
| $SO(3)$ | 3D Rotation Group |
| $\mathcal{U}(3)$ | Group of $3 \times 3$ Upper Triangular Matrices |

## 2. Magnetic Field Sensor's Error Sources and Measurement Model

In this section, the most important linear, time-invariant error sources of three-axis magnetic field sensors are presented. Based on them, a mathematical model relating the sensor's measurement with the actual value of the magnetic field is derived.

The total output error of a magnetic sensor is a combination of several error sources related to the sensing element itself, the instrumentation electronics, manufacturing imperfections and distortions caused by magnetic and ferromagnetic materials in the vicinity of the sensor. The linear, time-invariant error sources with the most significant contribution in the total sensor's error, are listed below:

- **Bias, or offset**; all magnetic sensors suffer from bias, which is a constant distortion. In many cases, it is the most important defect in the sensor's overall error. A $3 \times 1$ vector, $h_s$, is used to model it.

- **Scale-factor** error represents the input-output gain error of the sensor. It is modeled by a $3 \times 3$ diagonal matrix, $T_{sf}$.
- **Cross-coupling or non-orthogonality** inaccuracies are resulted by the non-ideal alignment of the sensor's axes during manufacturing and are modeled by a $3 \times 3$ matrix, $T_{cc}$.
- **Soft-iron distortion** is caused by ferromagnetic materials in the vicinity of the sensor, attached to the sensor's coordinate frame. Those materials do not generate their own magnetic field, but instead alter the existing magnetic field locally, resulting in a measurement discrepancy. This effect is modeled by a $3 \times 3$ matrix, $T_{si}$.
- **Hard-iron distortion** is due to magnetic materials attached to the sensor's coordinate frame. As a consequence of the persistent magnetic field created by those materials, the sensor's output has a constant bias. Hard-iron distortion is modeled by a $3 \times 1$ vector, $h_{hi}$.
- **Random noise** is the stochastic error in the sensor's output. It is induced by the sensor's mechanical and electrical architecture. It is modeled by a $3 \times 1$ vector, $\varepsilon$, and it is most commonly assumed to be a sequence of white noise, i.e., $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

Let $m$ be the $3 \times 1$ true magnetic field vector and $y$ be the $3 \times 1$ measurement vector. With the aforementioned error terms in mind, a widely accepted and well-referenced measurement model for a three-axis magnetometer is the following [1,2,4–7,13,14]

$$y = T_{sf} T_{cc} (T_{si} m + h_{hi}) + h_s + \varepsilon \tag{1}$$

In most applications, the exact contribution of each error term in (1) is of no concern and, thus, instead of (1), most calibration algorithms use the following, compact form of (1)

$$y = Tm + h + \varepsilon \tag{2}$$

where $T \triangleq T_{sf} T_{cc} T_{si}$ and $h \triangleq T_{sf} T_{cc} h_{hi} + h_s$.

This work focuses on algorithms intended to be used with magnetic field sensors requiring no special piece of equipment. In such cases, the calibration is done in the sensor's (body) coordinate frame implying that both the measurement vector, $y$ and the true magnetic field vector, $m$ in (2) are expressed in the senor's coordinate frame.

Note that when expensive laboratory equipment is not available, both the calibration parameters $T$ and $h$ in (2), and the magnetic field vector, $m$, are unknown. Thus, in most works, multiple measurements of the local (Earth's) magnetic field are used to derive $T$ and $h$. Note that the Earth's magnetic field varies with location and time and its value (magnitude and direction) is only approximately known by magnetic models such as International Geomagnetic Reference Field model (IGRF) [15]. However it is reasonable to assume that the magnitude of the magnetic field is (locally) constant during the calibration procedure. Based on this fact, most authors formulate an optimization or an estimation problem to derive $T$ and $h$.

## 3. Alonso and Shuster (TWOSTEP)

The TWOSTEP algorithm [1] consists of an analytical centering approach [16,17] for its first step, while in the second step the solution is optimized numerically. The authors initially solved the problem of bias, $h$, determination when attitude is not known [18] and then extended their method to determine matrix $T$ as well [1].

TWOSTEP is motivated by the assumption that matrix $T$ should not be far from a pure rotation. Therefore by applying polar decomposition it can be written as $T = (I_{3\times3} + D)^{-1} O$ where $O$ is an orthogonal matrix and $D$ is a symmetric $3 \times 3$ matrix so as $(I_{3\times3} + D)^{-1}$ to be positive definite. Matrix $O$ can be integrated into vector $m$ since it does not alter its norm. The equivalent measurement model is

$$y = \hat{T}\hat{m} + h + \varepsilon \tag{3}$$

where

$$\hat{T} \triangleq (I_{3\times3} + D)^{-1}$$
$$\hat{m} \triangleq Om.$$

Therefore, for the full calibration, $D$ and $h$ must be estimated. To this purpose, a set of $N$ measurements, $y_k$, $k = 1, 2, \ldots N$, is used and the corresponding effective measurements $z_k$, $k = 1, 2, \ldots N$, are defined as

$$\begin{aligned} z_k &\triangleq \|y_k\|^2 - \|\hat{m}_k\|^2 \\ &= \|y_k\|^2 - \|m_k\|^2. \end{aligned} \tag{4}$$

The last ones can be decomposed into a deterministic part plus an approximately Gaussian noise term, $v_k$ with mean $\mu_k$ and variance $\sigma_k^2$, i.e., $v_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$, given by

$$\mu_k = -3\sigma^2 \tag{5a}$$
$$\sigma_k^2 = 4\sigma^2((I_{3\times3} + D)y_k - h)^T((I_{3\times3} + D)y_k - h) + 6\sigma^4. \tag{5b}$$

Since $D$ and $h$ are unknown, the variance $\sigma_k^2$ is assumed to be similar to measurement's output error variance $\sigma^2$. Hence $\mu_k$ and $\sigma_k^2$ can be assumed independent of $k$. To estimate $D$ and $h$, Alonso and Shuster define the auxiliary quantities $E \triangleq D^2 + 2D$ and $c \triangleq (I + D)h$, as well as the estimation vector $\theta'$ containing the elements of the $3 \times 1$ vector $c$ and those of the $3 \times 3$ symmetric matrix $E$, which is formed as $\theta' = [c^T E_{11}\ E_{22}\ E_{33}\ E_{12}\ E_{13}\ E_{23}]^T$.

TWOSTEP algorithm functions on the estimation vector $\theta'$ and thus on the auxiliary parameters, $E$ and $c$ and not on the actual calibration parameters, $D$ and $h$. The transformation from $E$ and $c$ back to $D$ and $h$ is described in (13) and (14).

### *3.1. Initial Estimate*

For every measurement, $y_k$, $k = 1, 2, \ldots N$, the following auxiliary variables are defined

$$S_k = [y_{k,1}^2\ y_{k,2}^2\ y_{k,3}^2\ 2y_{k,1}y_{k,2}\ 2y_{k,1}y_{k,3}\ 2y_{k,2}y_{k,3}] \tag{6a}$$
$$L_k = [2y_k^T\ -S_k]. \tag{6b}$$

The centering approximation is done using the following weighted averages along with their corresponding centered values

$$\bar{z} \triangleq \bar{\sigma}^2 \sum_{k=1}^{N} \frac{1}{\sigma_k^2} z_k \qquad \tilde{z}_k \triangleq z_k - \bar{z} \tag{7a}$$
$$\bar{L} \triangleq \bar{\sigma}^2 \sum_{k=1}^{N} \frac{1}{\sigma_k^2} L_k \qquad \tilde{L}_k \triangleq L_k - \bar{L} \tag{7b}$$
$$\bar{\mu} \triangleq \bar{\sigma}^2 \sum_{k=1}^{N} \frac{1}{\sigma_k^2} \mu_k \qquad \tilde{\mu}_k \triangleq \mu_k - \bar{\mu} \tag{7c}$$

where

$$\bar{\sigma}^2 \triangleq \left( \sum_{k=1}^{N} \frac{1}{\sigma_k^2} \right)^{-1}.$$

The first estimation of $\theta'$ is the centered one given by

$$\tilde{\theta}' = \tilde{P}_{\theta'\theta'} \sum_{k=1}^{N} \frac{1}{\sigma_k^2} (\tilde{z}_k - \tilde{\mu}_k) \tilde{L}_k^T \tag{8a}$$

$$\tilde{P}_{\theta'\theta'}^{-1} = \sum_{k=1}^{N} \frac{1}{\sigma_k^2} \tilde{L}_k^T \tilde{L}_k \tag{8b}$$

with $\tilde{P}_{\theta'\theta'}$ denoting the centered covariance matrix and $\tilde{F}_{\theta'\theta'}$ denoting the centered Fischer information matrix.

### *3.2. Solution Improvement Step*

The second step improves the previous estimate of vector $\theta$, derived in (8), via Gauss-Newton method using the centered estimate $\tilde{\theta}'$ as the initial guess. The estimation is updated as follows

$$\theta'_{i+1} = \theta'_i - \left[ \tilde{F}_{\theta'\theta'} + \frac{1}{\bar{\sigma}^2} \left( \bar{L} - \phi(\theta'_i) \right)^T \left( \bar{L} - \phi(\theta'_i) \right) \right]^{-1} g(\theta'_i) \tag{9}$$

where

$$v = (I_{3\times 3} + E)^{-1} c \tag{10a}$$

$$\phi(\theta') = [2v^T \ -v_1^2 \ -v_2^2 \ -v_3^2 \ -2v_1v_2 \ -2v_1v_3 \ -2v_2v_3] \tag{10b}$$

$$g(\theta') = \tilde{P}_{\theta'\theta'}^{-1} (\theta' - \tilde{\theta}') - \frac{1}{\bar{\sigma}^2} \left( \bar{z} - \bar{L}\theta' + c^T v - \bar{\mu} \right) \left( \bar{L}^T - \phi(\theta') \right) \tag{10c}$$

with $v_j$ denoting the $j^{th}$ element of vector $v$. At every iteration the $3 \times 3$ symmetric matrix $E$ and the $3 \times 1$ vector $c$ are updated according to the current estimation vector $\theta'_i$ using

$$c = \begin{bmatrix} \theta'_1 \\ \theta'_2 \\ \theta'_3 \end{bmatrix} \quad \text{and} \quad E = \begin{bmatrix} \theta'_4 & \theta'_7 & \theta'_8 \\ \theta'_7 & \theta'_5 & \theta'_9 \\ \theta'_8 & \theta'_9 & \theta'_6 \end{bmatrix}. \tag{11}$$

Alonso and Shuster define the following quantity in order to establish a stop condition for the Gauss–Newton method.

$$\eta_i \triangleq \left( \theta'_{i+1} - \theta'_i \right)^T \left[ \tilde{F}_{\theta'\theta'} + \frac{1}{\bar{\sigma}^2} \left( \bar{L} - \phi(\theta'_i) \right)^T \left( \bar{L} - \phi(\theta'_i) \right) \right] \left( \theta'_{i+1} - \theta'_i \right). \tag{12}$$

The iterations continue until $\eta_i$ became smaller than a predetermined threshold.

After sufficiently many iterations, an optimal estimation of matrix $E^*$ and of vector $c^*$ is derived. The derived solution is then used to find $D^*$ and $h^*$. To this end we apply SVD [19] to the symmetric matrix $E^*$, i.e.,

$$E^* = USU^T \tag{13}$$

where $S = \text{diag}(s_1, s_2, s_3)$, $U \in O(3)$. Then find the diagonal matrix $W = \text{diag}(w_1, w_2, w_3)$ satisfying $S = 2W + W^2$. Typically, the elements of $S$ are much smaller than unity [1] so a real solution exists with the diagonal entries of $W$ being $w_j = -1 + \sqrt{1 + s_j}$, $j = 1, 2, 3$.

Combining the above, the estimates of matrix $D^*$ and bias vector $h^*$ are given by

$$D^* = UWU^T \tag{14a}$$

$$h^* = (I_{3\times 3} + D^*)^{-1} c^* \tag{14b}$$

and are related to the calibration parameters $T$ and $h$ of the measurement model (2) as follows

$$T = (I_{3\times3} + D^*)^{-1} \text{ and } h = h^*. \tag{15}$$

Summarizing, when the centered estimation is near the ground truth value the Gauss–Newton method typically converges rapidly. The authors verified the robustness of their method via simulations assuming either white noise or colored noise. TWOSTEP is also suitable for on-orbit calibration using IGRF data [15]. The algorithm is summarized in Algorithm 1.

**Algorithm 1:** Alonso and Shuster (TWOSTEP) [1]

Step 1: Calculate $z_k, L_k,$ for $k = 1, 2, \ldots, N$ by using (4)–(6)
Step 2: Calculate the centered values $\tilde{z}_k, \tilde{L}_k$ for $k = 1, 2, \ldots, N$ (7)
Step 3: Calculate centered estimate $\tilde{\theta}'$ and covariance matrix $\tilde{P}_{\theta'\theta'}$ (8)
Step 4: Extract $c$ and $E$ from $\theta'$ following (11)
Step 5: Calculate $\phi(\theta')$ and $g(\theta')$ from (10)
Step 6: Update $\theta'$ using (9)
Step 7: Calculate $\eta$ following (12)
Step 8: Repeat steps 4–7 until $\eta$ is sufficiently small
Step 9: Apply SVD on $E^*$ (13) and define matrix $W$
Step 10: Calculate $D^*, h^*$ (14) and $T, h$ (15)

## 4. Crassidis et al.

The authors of [6] were motivated by the fact that real-time applications demand real-time calibration methods. To this end, based on the problem formulation (3) established in TWOSTEP [1], Crassidis et al. formulate a real-time identification problem for the derivation of the calibration parameters $D$ and $h$ and solve it using the extended Kalman Filter (EKF) approach. Note that the authors of [6] have proposed two more algorithms dealing with the online calibration of a magnetic field sensor in [20,21]. However, in this work we focus on the EKF based one presented in [6] which is the most efficient and popular one.

Following the problem formulation of TWOSTEP, the bias vector $h$ and the symmetric matrix $D$ are desired. The estimation vector $\theta$ is defined differently and contains $h$ and $D$, structured as follows

$$\theta = [h^T \; D_{11} \; D_{22} \; D_{33} \; D_{12} \; D_{13} \; D_{23}]^T. \tag{16}$$

Because the vector $\theta$ is constant, the state model is given by $\dot{\theta} = 0$. The effective measurement is given by $z_k = \|y_k\|^2 - \|m_k\|^2$ (4) while the measurement's model is given by $z_k = \phi(\theta_k) + v_k$ where

$$\phi(\theta_k) = -y_k^T \left(2D_k + D_k^2\right) y_k + 2y_k^T (I_{3\times3} + D_k) h_k - \|h_k\|^2 \tag{17}$$

and effective measurement's noise $v_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$ follows (5). At each iteration $D_k$ and $h_k$ are extracted from $\theta_k$ according to (16). The propagation is as it follows

$$\theta_{k+1} = \theta_k + K_k[z_k - \phi(\theta_k)] \tag{18a}$$
$$P_{k+1} = [I_{9\times9} - K_k H(\theta_k)]P_k \tag{18b}$$
$$K_k = P_k H^T(\theta_k) \left[H(\theta_k) P_k H^T(\theta_k) + \sigma_k^2\right]^{-1} \tag{18c}$$

where $P_k$ is the covariance of the estimated parameters for $h$ and $D$ at step $k$. The matrix $H(\theta_k)$ is the linearization matrix of $\phi(\theta_k)$ and is defined as

$$H(\theta_k) = [2y_k^T(I_{3\times3} + D_k) - 2h_k^T \quad - S_kF_k + 2J_k] \tag{19}$$

where

$$S_k = [y_{k,1}^2 \;\; y_{k,2}^2 \;\; y_{k,3}^2 \;\; 2y_{k,1}y_{k,2} \;\; 2y_{k,1}y_{k,3} \;\; 2y_{k,2}y_{k,3}] \tag{20a}$$

$$J_k = [y_{k,1}h_{k,1} \;\; y_{k,2}h_{k,2} \;\; y_{k,3}h_{k,3} \;\; y_{k,1}h_{k,2} + y_{k,2}h_{k,1} \;\; y_{k,1}h_{k,3} + y_{k,3}h_{k,1} \;\; y_{k,2}h_{k,3} + y_{k,3}h_{k,2}] \tag{20b}$$

$$F_k = \begin{bmatrix} \Delta_1 & 0 & 0 & 2D_{k,12} & 2D_{k,13} & 0 \\ 0 & \Delta_2 & 0 & 2D_{k,12} & 0 & 2D_{k,23} \\ 0 & 0 & \Delta_3 & 0 & 2D_{k,13} & 2D_{k,23} \\ D_{k,12} & D_{k,12} & 0 & \Delta_4 & D_{k,23} & D_{k,13} \\ D_{k,13} & 0 & D_{k,13} & D_{k,23} & \Delta_5 & D_{k,12} \\ 0 & D_{k,23} & D_{k,23} & D_{k,13} & D_{k,12} & \Delta_6 \end{bmatrix} \tag{20c}$$

and

$$\begin{aligned} \Delta_1 &= 2(1 + D_{k,11}) \\ \Delta_2 &= 2(1 + D_{k,22}) \\ \Delta_3 &= 2(1 + D_{k,33}) \\ \Delta_4 &= 2 + D_{k,11} + D_{k,22} \\ \Delta_5 &= 2 + D_{k,11} + D_{k,33} \\ \Delta_6 &= 2 + D_{k,22} + D_{k,33}. \end{aligned} \tag{21}$$

The noise variance of the measurements, $\sigma_k^2$, can be assumed to be constant and equal to $\sigma^2$ as in TWOSTEP. Given a set of $N$ measurements, the EKF provides an optimal estimation vector $\theta^* = \theta_N$ from which an optimal vector $h^* = h_N$ and a matrix $D^* = D_N$ can be extracted according to (16). Therefore, the calibration parameters (2) are given by

$$T = (I_{3\times3} + D^*)^{-1} \text{ and } h = h^*. \tag{22}$$

Under certain conditions, e.g., fast changing data, this approach of sequential calibration may have some advantages in terms of computational complexity and adaptation. The algorithm is summarized in Algorithm 2.

**Algorithm 2:** Crassidis et al. [6]

Step 1: Initialize $\theta$ and $k = 0$

Step 2: **for each** measurement do:
- Calculate $z_k$ (4)
- Extract $D_k$ and $h_k$ from $\theta_k$ (16)
- Calculate $S_k$, $J_k$, $F_k$ (20) and $H(\theta_k)$ (19)
- Calculate Kalman Gain $K_k$ (18)
- Update estimation: $\theta_k \leftarrow \theta_{k+1}$
- Update covariance matrix: $P_k \leftarrow P_{k+1}$ (18)
- $k \leftarrow k + 1$

Step 3: Extract $D^*$ and $h^*$ from $\theta^*$ (16)

Step 4: Calculate $T$ and $h$ (22)

### 5. Dorveaux et al.

An iterative algorithm for the calibration of magnetic field sensors based on iterations of a least-squares problem is introduced in [3]. In the beginning of the algorithm, the measurements lie on an ellipsoid according to (2). In each iteration, the measurements move from the initial ellipsoid to the unit sphere, following a cost function minimization algorithm.

The authors in [3] use the following variation of the measurement model of (2)

$$m = Ay + B \tag{23}$$

where $A = T^{-1}$, $B = -T^{-1}h$ and the measurement noise, $\varepsilon$, is neglected.

The algorithm begins by considering an initial estimate of the magnetic field vectors, denoted by $\tilde{m}_k(0)$ and defined as

$$\tilde{m}_k(0) = y_k,\ k = 1, 2, \ldots, K. \tag{24}$$

In every iteration, the following cost function is formulated and minimized using the least squares method.

$$J(A, B, n) = \sum_{k=1}^{K} \left\| A\tilde{m}_k(n) + B - \frac{\tilde{m}_k(n)}{\|\tilde{m}_k(n)\|} \right\|^2 \tag{25}$$

where $n = 1, 2, \ldots, N$ denotes the $n^{th}$ iteration. Let $A_n$ and $B_n$ be the resulting matrices from the minimization of (25). Every iteration ends with using $A_n$ and $B_n$ to update the estimates of the magnetic field vectors as

$$\tilde{m}_k(n+1) = A_n\tilde{m}_k(n) + B_n,\ k = 1, 2, \ldots, K. \tag{26}$$

From (26) we can express the magnetic field estimates $\tilde{m}_k(n)$ using the measurement vectors $y_k$ as

$$\tilde{m}_k(n) = \tilde{A}_n y_k + \tilde{B}_n,\ k = 1, 2, \ldots, K \tag{27}$$

where $\tilde{A}_n$ and $\tilde{B}_n$ are iteratively defined as

$$\tilde{A}_n = A_n\tilde{A}_{n-1} \text{ and } \tilde{B}_n = A_n\tilde{B}_{n-1} + B_n. \tag{28}$$

To determine when the algorithm has reached an acceptable solution, we define the following cost

$$J_{stop}(A_n, B_n) = \|B_n\| + \|A_n - I_{3\times 3}\|. \tag{29}$$

The iterations stop when $J_{stop}$ is sufficiently small. Note that the original manuscript does not provide an explicit condition to stop iterations. However it is reasonable to terminate the algorithm when contribution of the updated $A_n$ and $B_n$ to the calibration parameters $\tilde{A}_n$ and $\tilde{B}_n$ is negligible (see (28)). The estimate $\tilde{m}(N)$ derived at the $N$th iteration represents the calibrated data and it is:

$$m_k = \tilde{m}_k(N),\ k = 1, 2, \ldots, K \tag{30}$$

The derived matrices $\tilde{A}_N$ and $\tilde{B}_N$ are related to the calibration parameters $T$ and $h$ of the measurement model (2) as follows

$$T = \tilde{A}_N^{-1} \text{ and } h = -\tilde{A}_N^{-1}\tilde{B}_N. \tag{31}$$

Finally, the estimates $\tilde{m}_k(N)$, $k = 1, 2, \ldots, K$, derived at the $N^{th}$ iteration represent the calibrated measurement vectors. The algorithm is summarized in Algorithm 3.

**Algorithm 3:** Dorveaux et al. [3]

Step 1: Initialize $\tilde{m}_k(0)$ using (24).
Step 2: Minimize (25) using least squares and derive $A_n$ and $B_n$.
Step 3: Use $A_n$ and $B_n$ to calculate $\tilde{m}_k(n+1)$ from (26).
Step 4: Calculate $\tilde{A}_n$ and $\tilde{B}_n$ using (28).
Step 5: Evaluate the cost function $J_{stop}(A_n, B_n)$ from (29).
Step 6: Repeat steps 2-5 until $J_{stop}$ is sufficiently small.
Step 7: Use $\tilde{A}_N$ and $\tilde{B}_N$ to calculate $T$ and $h$ using (31).

**6. Vasconcelos et al.**

The authors of [2] consider that magnetometers' measurements lie on a ellipsoid manifold following the measurement model (2). First, they derive an initial estimate of the calibration parameters $T$ and $h$ by finding the ellipsoid that fits best to the given data. Then, they use the measurement model of (2) to formulate a maximum likelihood estimation problem and derive an improved estimate of the calibration parameters $T$ and $h$.

From (2), the magnetic field vector is expressed as $m = T^{-1}(y - h) - T^{-1}\varepsilon$. Assuming that the magnitude of the magnetic field is constant during the calibration procedure we can write the following unconstrained optimization problem to derive $T$ and $h$

$$\underset{T,h}{\text{minimize}} \quad \sum_{k=1}^{K} \left( \frac{\|T^{-1}(y_k - h)\| - 1}{\sigma_k} \right)^2. \tag{32}$$

Here $\sigma_k$ denotes the standard deviation of the measurement noise in the $k^{th}$ measurement, assuming it is the same for all three axes and equal to $\sigma$. Without loss of generality, the magnitude of the magnetic field is assumed to be equal to one. A possible relaxation of this soft assumption is provided by Springmann [22] who addresses the problem of time-varying bias. To solve (32), the authors define the following cost function and then minimize it using the Newton's method

$$J(x) \triangleq \sum_{k=1}^{K} \left( \frac{\|\hat{T}(y_k - h)\| - 1}{\sigma_k} \right)^2 \tag{33}$$

where $\hat{T} = T^{-1}$ and

$$x = \begin{bmatrix} \text{vec}(\hat{T})^T & h^T \end{bmatrix}^T. \tag{34}$$

The vector $x$ is updated in every Newton's iteration as follows

$$x^{(+)} = x^{(-)} - \left[ \nabla^2 J(x) \Big|_{x=x^{(-)}} \right]^{-1} \nabla J(x) \Big|_{x=x^{(-)}} \tag{35}$$

where $\nabla J(x)$ is the gradient vector and $\nabla^2 J(x)$ is the Hessian matrix of the cost function. For both $\nabla J(x)$ and $\nabla^2 J(x)$, the authors in [2] provide analytical expressions which are presented in Appendix A.1.

*Initial Estimate*

Solving (32) using the Newton's method requires a good initial estimate of the calibration parameters, $\hat{T}$ and $h$. Vasconcelos et al. use a previous work on nonlinear estimators for strap-down magnetometers by Foster and Elkaim [23,24], to derive a good initial estimate. Solving the ellipsoid equation $\|m_k\| = \|T^{-1}(y_k - h)\| = 1$ for every $k$ is equivalent to solving the following pseudo-linear least squares estimation problem by re-arranging the terms as follows

$$Lp = b \tag{36}$$

where, by writing each measurement vector as $y_k = \begin{bmatrix} y_k^x & y_k^y & y_k^z \end{bmatrix}^T, k = 1, 2, \ldots, K$, it is

$$L = \begin{bmatrix} y_1^{x\,2} & y_1^x y_1^y & y_1^x y_1^z & y_1^{y\,2} & y_1^y y_1^z & y_1^x & y_1^y & y_1^z & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_K^{x\,2} & y_K^x y_K^y & y_K^x y_K^z & y_K^{y\,2} & y_K^y y_K^z & y_K^x & y_K^y & y_K^z & 1 \end{bmatrix} \tag{37}$$

and

$$b = \begin{bmatrix} y_1^{z\,2} & y_2^{z\,2} & \ldots & y_K^{z\,2} \end{bmatrix}^T. \tag{38}$$

The vector $p$ is derived as

$$p = \begin{bmatrix} A & B & C & D & E & G & H & I & J \end{bmatrix}^T = (L^T L)^{-1} L^T b. \tag{39}$$

The initial estimates of the calibration parameters are derived as

$$\hat{T}(0) = \begin{bmatrix} \frac{1}{\alpha} & 0 & 0 \\ -\frac{1}{\alpha}\tan(\rho) & -\frac{1}{b}\sec(\rho) & 0 \\ \frac{1}{\alpha}(\tan(\rho)\tan(\lambda)\sec(\phi) - \tan(\phi)) & -\frac{1}{b}\sec(\rho)\tan(\lambda)\sec(\phi) & \frac{1}{c}\sec(\lambda)\sec(\phi) \end{bmatrix} \tag{40}$$

and

$$h(0) = \frac{1}{2\alpha_1}\begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix}^T \tag{41}$$

where

$$\begin{aligned} a &= \frac{1}{2\alpha_1}\left(-(4D + E^2)\alpha_2\right)^{1/2} \\ b &= \frac{1}{2\alpha_1}\left(-(4A + C^2)\alpha_2\right)^{1/2} \\ c &= \frac{1}{2\alpha_1}\left((4DA - B^2)\alpha_2\right)^{1/2} \\ \tan(\rho) &= -\frac{1}{2\alpha_1}(2B + EC)(\alpha_1)^{-1/2} \\ \tan(\phi) &= (BE - 2CD)(\alpha_1)^{-1/2} \\ \tan(\lambda) &= E(-\alpha_1\alpha_3^{-1})^{1/2} \end{aligned} \tag{42}$$

and

$$\begin{aligned} \beta_1 &= 2BH + BEI - 2CDI - 4DG + ECH - E^2G \\ \beta_2 &= -2AEI + 4AH - BCI - 2BG + C^2H - CEG \\ \beta_3 &= 4DIA - 2DGC + EGB - IB^2 - 2EHA + CBH. \end{aligned} \tag{43}$$

The auxiliary variables $\alpha_1$, $\alpha_2$ and $\alpha_3$ are defined as

$$\begin{aligned} \alpha_1 &= -B^2 + DC^2 + 4DA + AE^2 - BEC \\ \alpha_2 &= 4AE^2J - E^2G^2 - 4BECJ + 2ECHG + 2BEIG - 4EHAI - 4DICG - C^2H^2 \\ &\quad + 4DAI^2 + 2CBHI - 4DG^2 + 4DC^2J + 4BHG - 4AH^2 - B^2I^2 - 4B^2J + 16DAJ \\ \alpha_3 &= E^4A - CBE^3 + E^2C^2D - 2B^2E^2 + 8DAE^2 - 4DB^2 + 16D^2A. \end{aligned} \tag{44}$$

One contribution of Vasconcelos et al., advancing the existing initial step approach suggested in [23], was the derivation of the aforementioned explicit and non-trivial expressions. In addition, Vasconcelos et al. state that their proposed algorithm is applicable even when the magnitude of the magnetic field is not constant during the measurement, similarly to TWOSTEP and Crassidis et al. algorithm [6]. The algorithm is summarized in Algorithm 4.

**Algorithm 4:** Vasconcelos et al. [2]

**Initial Estimate**

Step 1: Use the sensors' measurements $y_k$, $k = 1, 2, \ldots, K$ and form $A$ and $b$ according to (37) and (38), respectively.
Step 2: Calculate $p$ using (39)
Step 3: Derive the initial estimates $\hat{T}(0)$ and $h(0)$ using (40) and (41), respectively.

**Newton Method**

Step 4: Use the initial estimates $\hat{T}(0)$ and $h(0)$ to initialize $x$ according to (34).
Step 5: Update $x$ using (35).
Step 6: Evaluate the cost function $J(x)$ of (33).
Step 7: Repeat Steps 5–6 until $J(x)$ becomes sufficiently small.
Step 8: Split $x$ into $\hat{T}$ and $h$ and calculate $T = \hat{T}^{-1}$.

**7. Ali et al.**

The authors in [7] propose a Particle Swarm Optimization (PSO) [25] - based calibration algorithm that estimates the bias, the scale and nonorthogonality factors. The main advantage of this algorithm is its simplicity of implementation since the optimization is heuristic and does not depend on calculation of gradients, unlike other optimization techniques mentioned in this paper. It can be classified as an AI [26] approach.

The authors in [7] use (2) and a set of $N$ sensor's measurements to form the following optimization problem for deriving the calibration parameters $T$ and $h$

$$\underset{T,h}{\text{minimize}} \sqrt{\sum_{k=0}^{N} \left( \|y_k\|^2 - \|m_k\|^2 \right)^2}. \tag{45}$$

where $J \triangleq \sqrt{\sum_{k=0}^{N} \left( \|y_k\|^2 - \|m_k\|^2 \right)^2}$ is called the fitness value.

Function $J$ depends on $T$ and $h$ which are conveniently combined into the single vector $x \in \mathbb{R}^{12}$,

$$x = \begin{bmatrix} h \\ \text{vec}(T^T) \end{bmatrix}. \tag{46}$$

For a swarm of $S$ particles, the position $x_i \in \mathbb{R}^{12}$ and the velocity $v_i \in \mathbb{R}^{12}$ of the $i^{th}$ particle can be computed using [25]

$$v_i^k = v_i^{k-1} + c_1 r_{1i}^{k-1} \left( p_i^{k-1} - x_i^{k-1} \right) + c_2 r_{2i}^{k-1} \left( p_g^{k-1} - x_i^{k-1} \right) \tag{47a}$$

$$x_i^k = x_i^{k-1} + v_i^k \tag{47b}$$

for $i = 1, 2, \ldots, S$ where $k$ denotes the new value while $k-1$ the old value. Also $p_i$ denotes the $i^{th}$'s particle best position, $p_g$ denotes the swarm's best position, $c_1$ and $c_2$ are the acceleration coefficients, $w$ is the inertial weight factor and $r_{1i}, r_{2i}$ are random numbers uniformly distributed within the range $[0, 1]$. Typical values of these quantities are $c_1 = c_2 = 2$, $w = 1$ and the number of particles $S$ is usually between 20 and 65.

Therefore, at each iteration $k$, each particle's fitness value $J(x_i^k)$ is calculated and quantities $p_i$ and $p_g$ are updated accordingly. The authors suggest three different stop criteria. Specifically, the iterations stop either when the fitness value $J$ of a particle is smaller than a predetermined threshold, or after a maximum number of iterations, or when the change of $J$ becomes insignificant with iterations. Upon termination of the algorithm, parameters $T$ and $h$ (2) are extracted from the swarms's optimal solution $p_g$ according to

$$\begin{bmatrix} h \\ \text{vec}(T^T) \end{bmatrix} = p_g. \tag{48}$$

Following the general concept of applying AI optimization algorithms, as was introduced in [7], one can also consider using more modern versions of the standard PSO, e.g., [27–29]. They are typically found as built-in functions in computational suites such as MATLAB [30]. The algorithm is summarized in Algorithm 5.

**Algorithm 5:** Ali et al. [7]

Step 1: Initialize $x_i, v_i$ for $i = 1, 2, \ldots, S$
and set $p_i = x_i$

Step 2: Find $j = \{i | i = 1, 2, \ldots, S$ and $J(p_i) \leftarrow \min\}$
Particle $i$ best: $J^i_{min} \leftarrow J(p_i)$
Global best: $p_g \leftarrow p_j$ and $J_{min} \leftarrow J(p_j)$

Step 3: **for each** particle $i$ do
Update $x_i, v_i$ (47)
Calculate $J(x_i)$ (45)
**if** $J(x_i) < J^i_{min}$
$J^i_{min} \leftarrow J(x_i)$ and $p_i \leftarrow x_i$
**if** $J(x_i) < J_{min}$
$J_{min} \leftarrow J(x_i)$ and $p_g \leftarrow x_i$

Step 4: Repeat Step 3 until an exit condition is met

Step 5: Extract $T$ and $h$ from $p_g$ (48)

## 8. Wu and Shi

The authors of [4], formulate the calibration of a three-axis magnetometer as a maximum likelihood estimation problem which is solved using the Gauss-Newton method.

Starting from the measurement model of (2), Wu and Shi observed that by considering the QR decomposition $T^{-1} = QR$, where $Q \in O(3)$ and $R \in U(3)$, (2) is written as

$$y = R^{-1}Q^T m + h + \varepsilon. \tag{49}$$

Defining $\hat{m} \triangleq Q^T m$, we observe that $\|\hat{m}\| = \|m\|$ since $Q \in O(3)$. Also setting $\hat{T} \triangleq R^{-1}$ we have that

$$y = \hat{T}\hat{m} + h + \varepsilon. \tag{50}$$

Using the above transformation, the authors reduce the unknown model parameter variables from 12 (9 for $T$ and 3 for $h$) to 9 (6 for $R$ since $R$ is upper triangular and 3 for $h$). Note that using (50), the calibration procedure now aims at finding the calibration parameters $\hat{T}$ and $h$ while the magnetic field vector $\hat{m}$ is also unknown.

Using a set of $K$ measurements and (50), the authors formulate the following maximum likelihood estimation problem

$$\begin{aligned} \underset{\hat{T}, h, \hat{m}_k}{\text{minimize}} \quad & \sum_{k=1}^{K} \|y_k - \hat{T}\hat{m}_k - h\|^2 \\ \text{subject to} \quad & \|\hat{m}_k\| = 1, \; k = 1, 2, \ldots, K. \end{aligned} \tag{51}$$

Without loss of generality, the authors, constrained the magnitude of the magnetic field to be equal to one. Based on (51), the following Lagrange function is formulated

$$J(x) = \sum_{k=1}^{K} \left[ \|y_k - \hat{T}\hat{m}_k - h\|^2 + \lambda_k \Big( \|\hat{m}_k\|^2 - 1 \Big) \right] \tag{52}$$

where

$$x = \left[ \text{vec}(\hat{T})^T, \; h^T, \; \hat{m}_1^T, \; \hat{m}_2^T, \; \ldots, \; \hat{m}_K^T, \; \lambda_1, \; \lambda_2, \; \ldots, \; \lambda_K \right]^T \tag{53}$$

and $\lambda_k, k = 1, 2, \ldots, K$ are positive Lagrange coefficients for the unit norm constrain. Note that since $\hat{T}$ is an upper triangular matrix, the lower triangular elements of $\hat{T}$ are excluded

from $x$. The minimization of (52) and the estimation of $x$ are done using the Gauss-Newton method as follows

$$x^{(+)} = x^{(-)} - \left[\nabla^2 J(x)\Big|_{x=x^{(-)}}\right]^{-1}\left(\nabla J(x)\Big|_{x=x^{(-)}}\right) \tag{54}$$

where $\nabla J(x)$ is the Jacobian vector and $\nabla^2 J(x)$ is the Hessian matrix of the Lagrange function. For both $\nabla J(x)$ and $\nabla^2 J(x)$, the authors provide analytical expressions which are presented in Appendix A.2.

*Initial Estimate*

Solving (51) using the Gauss-Newton method requires a good initial estimate of the unknowns. To find one, the authors of [4] use the unit magnitude constrain and the equation $1 = \|R(y_k - h)\|^2$ which after some manipulation, is written as

$$\begin{bmatrix} y_k^T \otimes y_k^T & y_k^T & 1 \end{bmatrix} \begin{bmatrix} \mathrm{vec}(A) \\ b \\ c \end{bmatrix} \triangleq Y_k z = 0,\ k = 1, 2, \ldots, K \tag{55}$$

where $A = R^T R$, $b = -2R^T Rh$ and $c = h^T R^T Rh$. Defining $Y = \begin{bmatrix} Y_1^T & Y_2^T & \ldots & Y_K^T \end{bmatrix}^T$, from (55) it is

$$Yz = 0 \tag{56}$$

The authors, solve (56) using the least squares method and denote the solution $z_e = \begin{bmatrix} \mathrm{vec}(A_e)^T & b_e^T & c_e \end{bmatrix}^T = \min \|Yz\|^2$. They derive $z_e$ as the eigenvector of $Y^T Y$ corresponding to its minimum (or zero) eigenvalue. Using $z_e$, the vector $z$ is derived as $z = \alpha z_e$, where $\alpha = 4/\left(b_e^T A_e^{-1} b_e - 4c_e\right)$. Extracting $\mathrm{vec}(A)$, $b$ and $c$ from $z$, the initial estimates of the unknowns, $\hat{T}(0), h(0), \hat{m}_k(0)$ and $\lambda_k(0)$ are defined as follows:

$$\begin{aligned} \hat{T}(0) &= R^{-1} = \mathrm{chol}(A) \\ h(0) &= -A^{-1} b/2 \\ \hat{m}_k(0) &= \hat{T}(0)^{-1}(y_k - h),\ k = 1, 2, \ldots, K \\ \lambda_k(0) &= 0,\ k = 1, 2, \ldots, K \end{aligned} \tag{57}$$

where $\mathrm{chol}(\cdot)$ is the Cholesky factorization.

Finally, an alternative version of Wu's and Shi's algorithm is proposed by Cao et al. in [13], where a different method for the initial estimate is presented, and the second step is identical. The algorithm is summarized in Algorithm 6.

**Algorithm 6:** Wu and Shi [4]

**Initial Estimate**

Step 1: Calculate $Y_k$, $k = 1, 2, \ldots, K$ from (55) and form the matrix $Y = \begin{bmatrix} Y_1^T & Y_2^T & \ldots & Y_K^T \end{bmatrix}^T$.

Step 2: Find the eigenvector of $Y^T Y$ corresponding to its minimum (or zero) eigenvalue and denote it as $z_e = \begin{bmatrix} \mathrm{vec}(A_e)^T & b_e^T & c_e \end{bmatrix}^T$.

Step 3: Calculate $z = az_e$ where $\alpha = 4/\left(b_e^T A_e^{-1} b_e - 4c_e\right)$.

Step 4: Extract $\mathrm{vec}(A)$, $b$ and $c$ from $z$.

Step 5: Calculate an initial estimate of the unknowns using (57).

**Gauss–Newton Method**

Step 6: Use the initial estimates to initialize the vector $x$ of (53)

Step 7: Update $x$ using (54).

Step 8: Evaluate the cost $J(x)$ of (52).

Step 9: Repeat steps 7-8 until $J(x)$ becomes sufficiently small.

**9. Papafotis and Sotiriadis (MAG.I.C.AL.)**

The authors in [5] use (2) and a set of $K$ sensor's measurements to form the following optimization problem for deriving the calibration parameters $T$ and $h$

$$\begin{aligned} \underset{T,h,m_k}{\text{minimize}} \quad & \sum_{k=1}^{K} \|y_k - Tm_k - h\|^2 \\ \text{subject to} \quad & \|m_k\| = 1,\ k = 1, 2, \ldots, K \end{aligned} \tag{58}$$

where, without loss of generality, the magnitude of the magnetic field is constrained to be equal to one. In order to solve (58) they propose an iterative algorithm, based on the solution of a linear least-squares problem.

The algorithm begins by initializing the magnetic field vectors, $m_k$, as

$$m_k = \frac{y_k}{\|y_k\|},\ k = 1, 2, \ldots, K \tag{59}$$

and rewriting (2) in a matrix form as follows:

$$Y = LG + E \tag{60}$$

where

$$Y = \begin{bmatrix} y_1 & y_2 & \ldots & y_K \end{bmatrix} \tag{61a}$$

$$L = \begin{bmatrix} T & h \end{bmatrix} \tag{61b}$$

$$G = \begin{bmatrix} m_1 & m_2 & \ldots & m_K \\ 1 & 1 & \ldots & 1 \end{bmatrix} \tag{61c}$$

$$E = \begin{bmatrix} \varepsilon_1 & \varepsilon_2 & \ldots & \varepsilon_K \end{bmatrix}. \tag{61d}$$

In every iteration, (60) is solved for $L$ using the least squares method, minimizing the total squared error $\|E^T E\|_F^2$

$$L = YG^T(GG^T)^{-1} \tag{62}$$

From the calculated $L$, an updated set of calibration parameters $T$ and $h$ is extracted from (61b). Using them, the magnetic field vector is updated as

$$m_k = \frac{\tilde{m}_k}{\|\tilde{m}_k\|},\ k = 1, 2, \ldots, K \tag{63}$$

where

$$\tilde{m}_k = T^{-1}(y_k - h),\ k = 1, 2, \ldots, K. \tag{64}$$

Every iteration ends by updating the matrix $G$ using the updated vectors $m_k$, $k = 1, 2, \ldots, K$. Iterations stop when a small value of the following cost function is achieved

$$J(T, h) = \sum_{k=1}^{K} \left( \|\tilde{m}_k\|^2 - 1 \right)^2. \tag{65}$$

MAG.I.C.AL. algorithm is summarized in Algorithm 7.

**Algorithm 7:** Papafotis and Sotiriadis (MAG.I.C.AL.) [5]

Step 1: Initialize $m_k$ using (59).
Step 2: Calculate $L$ using (62).
Step 3: Extract $T$ and $h$ from $L$ using (61b).
Step 4: Update $m_k$ using (63) and (64) and use it to update $G$.
Step 5: Evaluate the cost-plus-penalty function $J$ from (65).
Step 6: Repeat steps 2-5 until $J(T, h)$ is sufficiently small.

## 10. Algorithm Evaluation and Comparison

In this Section, the performance of the presented algorithms are evaluated in terms of accuracy, robustness, and execution speed. Firstly, we evaluate the performance of the seven algorithms using multiple sets of synthetic data where the calibration parameters $T$ and $h$, as well as the measurement noise characteristics are predefined and known. By doing so, we are able to accurately determine the algorithms' accuracy and robustness. Then multiple datasets of two different low-cost magnetic field sensors are used to verify the algorithms' performance under real-world conditions.

### *10.1. Synthetic Data Generation*

We designed a procedure to generate synthetic data effectively, in order to examine each of the aforementioned algorithm's performance across a range of noise variance and measurement sample size. The authors of TWOSTEP [18] propose a typical scenario of assuming the magnetic vector spinning with a constant angular velocity. On the other hand, Wu and Shi [4] suggest a specific sequence of 3D rotations using Euler Angles, applied on a constant known magnetic vector $m$. In the same page, Papafotis and Sotiriadis [5] recommend a sequence of 12 approximate orientations. Another alternative is to make use of a set of random, yet normalized, vector fields, which however demands a reasonable amount of samples.

Because none of the described algorithms guarantees that it will function properly under an arbitrary dataset, we propose an efficient method to span $SO(3)$, following [31], so as to provide the algorithms with substantial, non-redundant information and to compare them fairly. After extensive simulation, it was observed that the recommended method was very effective in spanning the 3D rotation space.

Our method's effectiveness lies in distributing the points on the sphere $\|m\| = 1$, more evenly by using the canonical Fibonacci lattice mapping [31,32]. Generating a Fibonacci sphere is an extremely fast and effective approximate method to evenly distribute points on a sphere.

This way $SO(3)$ is sufficiently represented even with only a small dataset. An algorithm for generating $K$ vectors distributed on a Fibonacci sphere is presented in detail in Algorithm 8.

Considering $K$ vectors, $m_k$, $k = 1, 2, \ldots, K$ distributed on a Fibonacci sphere, we continue with generating matrix $T$ and vector $h$, required to calculate the corresponding measurement vectors $y_k$, $m_k$, $k = 1, 2, \ldots, K$ according to (2). Ideally, matrix $T$ would be the $3 \times 3$ identity matrix while the bias vector $h$ would be the $3 \times 1$ vector of zeros. A realistic model for $T$ and $h$, accounting for the sensor's non-idealities, is derived by using the concept of additive perturbation

$$T = \alpha I_{3\times 3} + E \tag{66a}$$

$$h = e \tag{66b}$$

where $\alpha$ accounts for gross scaling errors, $E$ is a $3 \times 3$ perturbation matrix with random, typically small, coefficients and $e$ is $3 \times 1$ perturbation bias vector with random coefficients.

Finally, a sequence of white noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is added to the measurements and the measurement vectors $y_k$, $m_k$, $k = 1, 2, \ldots, K$ are derived according to (2)

$$y = Tm + h + \varepsilon. \tag{67}$$

**Algorithm 8:** Generation of synthetic data

Step 1:Initialize the number of measurements $K$ and the radius of sphere $r$

Step 2: Calculate Golden Ratio: $\varphi = \frac{1+\sqrt{5}}{2}$

Step 3: **for each** $k = 1, 2, \ldots, K$ do:

$\theta = \frac{2\pi k}{\varphi}$

$\phi = \arccos\left(1 - \frac{2(k-0.5)}{K}\right)$

$m_k = \left[m_x, m_y, m_z\right] = [r\cos\theta\sin\phi, r\sin\theta\sin\phi, r\cos\phi]$

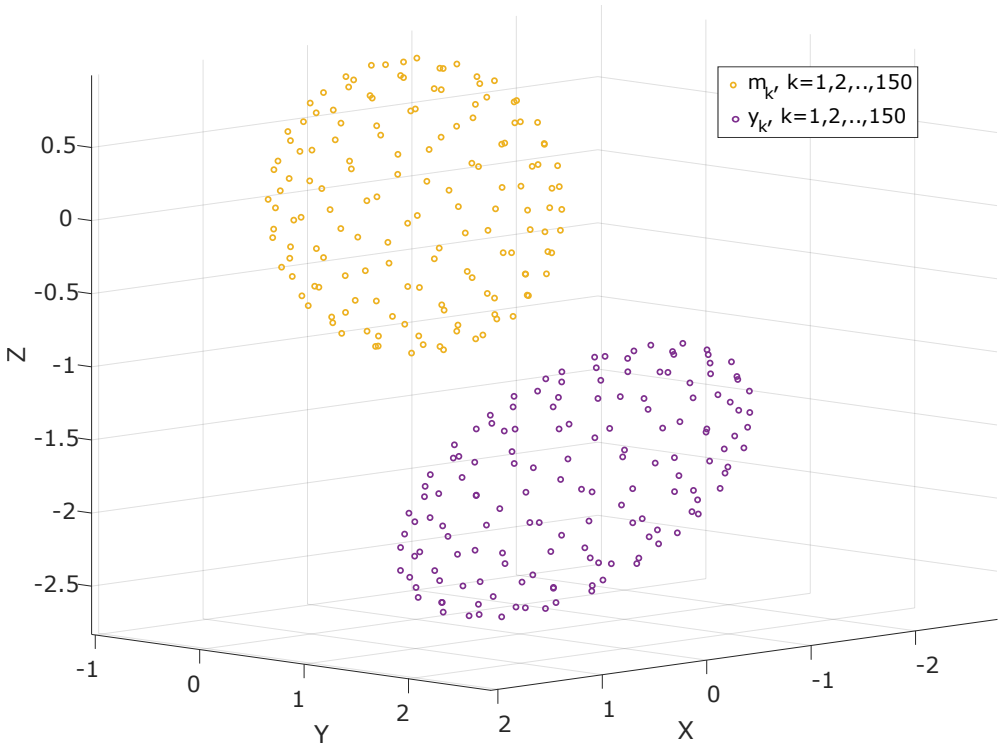
Step 4: Pick the scaling parameter, $\alpha$, the perturbation matrix, $E$ and the perturbation vector, $e$.

Step 5: Calculate $T$ and $h$ according to (66).

Step 6: Generate a sequence of white noise: $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

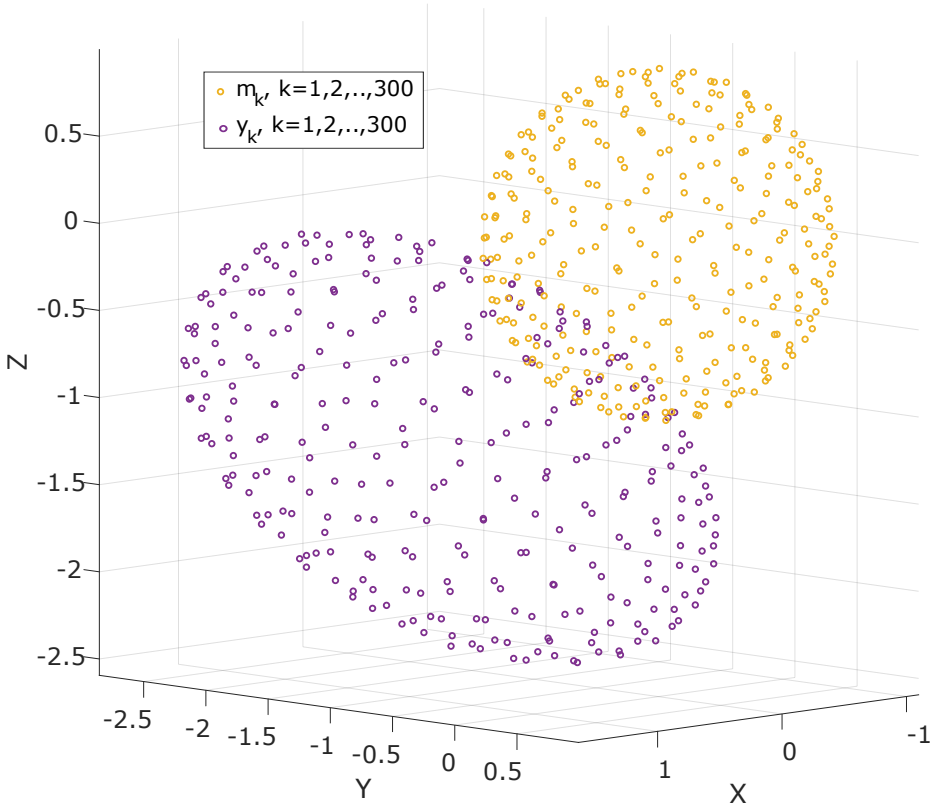
Step 7: Calculate the measurement vectors: $y_k = Tm_k + h + \varepsilon_k$ (2)

The two datasets generated using Algorithm 8 are presented in Figure 1. Note that for visualization purposes, the scaling parameter, $\alpha$, the perturbation matrix, $E$, and the perturbation vector, $e$, used to create each dataset were set to a rather large value.




(**a**) Synthetic dataset generated using Algorithm 8 for $K = 150$.

**Figure 1.** *Cont.*




(**b**) Synthetic dataset generated using Algorithm 8 for $K = 300$.

**Figure 1.** Two synthetic datasets generated using Algorithm 8 for $K = 150$ (**a**) and $K = 300$ (**b**), respectively.

#### 10.1.1. Experiment Setup and Evaluation Criteria

To evaluate the performance of the algorithms, we used synthetic data, generated by Algorithm 8, and we executed a great number Monte Carlo simulations. Each simulation consisted of 250 runs of each algorithm while in each run, the same dataset was used as input in all algorithms. An uncertainty was introduced in the generation of each dataset by considering a statistical distribution for the elements, $E_{ij}$, of the perturbation matrix, $E$, and the elements, $e_i$, of the perturbation vector $e$ (see (66)). Specifically, for the Monte Carlo simulations we assumed

$$\alpha \sim \mathcal{U}[0.8, 1.2] \tag{68a}$$

$$E_{ij} \sim \mathcal{U}[-\beta, \beta] \tag{68b}$$

$$e_i \sim \mathcal{U}[-\gamma, \gamma] \tag{68c}$$

where $\beta$ and $\gamma$ are scalars, the effect of which was tested using multiple Monte Carlo simulations. Note that we considered the scaling factor, $\alpha$, to be close to the ideal value of $\alpha = 1$. That may not be the case when real-world measurements are used, however, it is trivial, and common, to properly scale the measurements before the calibration procedure and remove gross scaling errors. In this way, the algorithms are not burdened, searching for a scaling relationship which can be easily provided by simple data preprocessing.

A challenging point while setting up the experiments was to determine the number of samples of each dataset and the value of the sensor's noise variance, $\sigma^2$. We considered a dataset of 300 measurements as a solid choice for a simulation environment based on [4,7] while we experimentally confirmed that bigger datasets do not improve the performance of any algorithm. We also examined the performance of the presented algorithms when smaller datasets, consisting of 150 and 50 measurements, are used. As far as the noise vari-

ance, $\sigma^2$, is concerned, we considered a nominal value of $\sigma = 0.005$, following [2,4], while we also simulated the cases of more noisy ($\sigma = 0.05$) and less noisy ($\sigma = 0.0005$) sensors.

The evaluation of the algorithm for each Monte Carlo simulation was completed in terms of accuracy, execution speed, and robustness. We used the execution speed of each algorithm as a metric of computational efficiency and is defined as the inverse of the mean execution time. As a metric of robustness we considered the percentage of datasets for which each algorithm successfully derived a meaningful solution.

The definition of an accuracy metric is a little more involved. Each algorithm was developed to take as inputs the measurement vectors $y_k$, $k = 1, 2, \ldots, K$ and output the calibration parameters $T$ and $h$. Comparing the output bias vector $h$ with the true one, $h_{true}$, which was used in the data generation procedure, was performed by defining the following cost

$$J_h = \|h_{true} - h\|. \tag{69}$$

The calibration matrix $T$ on the other hand is derived under a rotational uncertainty and comparing it with the true one, $T_{true}$, is a more challenging task.

Consider the measurement model of (2). Noting that the true magnetic field vector in (2) is also unknown, and derived by the calibration algorithm, we can write:

$$y = T_{true} R R^T m + h_{true} \tag{70}$$

where $R$ is an orthogonal matrix in the $O(3)$ group. Thus, taking into account the rotational invariance of the Euclidean norm which implies that $\|R^T m\| = \|m\|$, a calibration algorithm may output any matrix $T$ of the form $T = T_{true} R$. Thus, a proper cost function to compare $T$ and $T_{true}$ is the following

$$J_T = \|T - T_{true} R\|_F \tag{71}$$

where, the matrix $R$ is defined as the solution of the following minimization problem

$$R = \underset{\Omega \in O(3)}{\operatorname{argmin}} \|T - T_{true} \Omega\|_F. \tag{72}$$

The solution of (72) is given by the orthogonal procrustes problem [33], and it is

$$R = U V^T \tag{73}$$

where the matrices $U$ and $V$ are derived from the singular value decomposition (SVD) of the matrix $T_{true}^T T$, i.e., $T_{true}^T T = U \Sigma V^T$, where $U, V \in O(3)$ and $\Sigma$ is a diagonal matrix.

Using (69) and (71) we define the following cost function as a metric of accuracy

$$J = \|h_{true} - h\| + \|T - T_{true} R\|_F. \tag{74}$$

Based on the above and given the results of a Monte Carlo simulation consisted of $N$ executions of each algorithm, we define the following metrics of performance:

- **Accuracy** is defined as the mean value of the cost $J$, defined in (74), across all $N$ executions with meaningful output.
- **Mean execution time** is defined as the mean value of the execution time of an algorithm.
- **Robustness** is defined as the percentage of datasets for which each algorithm successfully derived a meaningful solution.

The robustness criterion can be seen as the frequency in which an algorithm provides a better solution $(T, h)$ in the sense of the cost function (74), than the trivial solution $(I_{3\times3}, 0_{3\times1})$ which assumes no bias and non multiplicative errors. Given the cost $J_o$ that corresponds to the trivial solution,

$$J_o = \|h_{true} - 0_{3\times1}\| + \|I_{3\times3} - T_{true} R\|_F \tag{75}$$

an execution of an algorithm is considered as successful with meaningful output when

$$J < \delta J_o \tag{76}$$

where $\delta \in (0,1)$ is a robustness parameter. If $\delta$ is close to 1, it means that only little improvement with respect to $J_o$ is sufficient. As $\delta$ gets smaller, better solutions are required. Thus, this parameter can be tuned with respect to the test's objective and the application's specifications. Given $N$ runs for an algorithm, its robustness is denoted by $RB(\%)$ and is defined as

$$RB(\%) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{U}(J_i < \delta J_{oi}) \cdot 100. \tag{77}$$

Here $J_i$ and $J_{oi}$ are the values of $J$ (74) and $J_o$ (75), respectively, corresponding to the $i$th run of the algorithm and $\mathbb{U}$ is a boolean function, which is one if its argument is true and zero otherwise. Let $M$ denote the number of executions meaningful outputs.

Now, the accuracy metric is only applied on the $M$ meaningful outputs according to the robustness test (76), since otherwise the comparison would be unfair for the least stable algorithms. The accuracy of an algorithm over a dataset is denoted by $\rho$ and it is defined as

$$\rho = \frac{1}{M}\sum_{i=1}^{N}\mathbb{U}(J_i < \delta J_{oi})J_i \tag{78}$$

which is the mean accuracy metric value over the $M$ executions with meaningful outputs.

Similarly, the time-efficiency metric (i.e., mean execution time) is only applied on the $M$ executions with meaningful outputs according to the robustness test (76). Again, this is because otherwise the comparison would be unfair for the least stable algorithms. The mean execution time of an algorithm over a dataset, is denoted by $\tau$ and is defined as

$$\tau = \frac{1}{M}\sum_{i=1}^{N}\mathbb{U}(J_i < \delta J_{oi})t_i \tag{79}$$

where $t_i$ is the time needed for the $i$ run to be completed. The execution speed of an algorithm is defined as $1/\tau$.

#### 10.1.2. Baseline Evaluation

To derive a baseline evaluation of the presented algorithms, we run a Monte Carlo simulation considering typical values for the sensor's error and noise parameters. In this simulation we neglected the effect of hard-iron and soft-iron distortions which are, in some cases, the dominant terms of the overall error, as well as extreme cases of large manufacturing imperfections. More specifically, 250 different datasets consisting of 300 measurements each, were generated following Algorithm 8 and considering the following distributions of the model disturbances and the measurement noise

$$\begin{aligned} \alpha &\sim \mathcal{U}[0.8, 1.2] \\ E_{ij} &\sim \mathcal{U}[-0.05, 0.05] \\ e_i &\sim \mathcal{U}[-0.05, 0.05] \\ \sigma &= 0.005 \end{aligned} \tag{80}$$

The distribution ranges in (80) are based on our literature review. The selection $\beta = \gamma = 0.05$ corresponds to the typical case of approximately 5% distortion for $T$ and bias $h$. The measurement noise standard deviation is set to a typical value of $\sigma = 0.005$ [2,4].

The performance of the seven algorithms is presented in Table 2.

**Table 2.** Baseline evaluation of the presented algorithms.

| Algorithm | Accuracy ($1/\rho$) | Robustness ($RB\%$) | Execution Speed ($1/\tau$) |
|---|---|---|---|
| TWOSTEP [1] | $35.3 \times 10^0$ | 91.6% | 455 $s^{-1}$ |
| Crassidis et al. [6] | $3.31 \times 10^3$ | 100% | 47.6 $s^{-1}$ |
| Dorveaux et al. [3] | $2.26 \times 10^5$ | 100% | 12.8 $s^{-1}$ |
| Vasconcelos et al. [2] | $2.28 \times 10^5$ | 99.6% | 0.089 $s^{-1}$ |
| Ali et al. [7] | $2.27 \times 10^5$ | 98.8% | 0.10 $s^{-1}$ |
| Wu and Shi [4] | $2.32 \times 10^5$ | 87.2% | 0.24 $s^{-1}$ |
| MAG.I.C.AL [5] | $2.28 \times 10^5$ | 100% | 29.4 $s^{-1}$ |

10.1.3. The Effect of the Offset Perturbation Parameter, $\gamma$

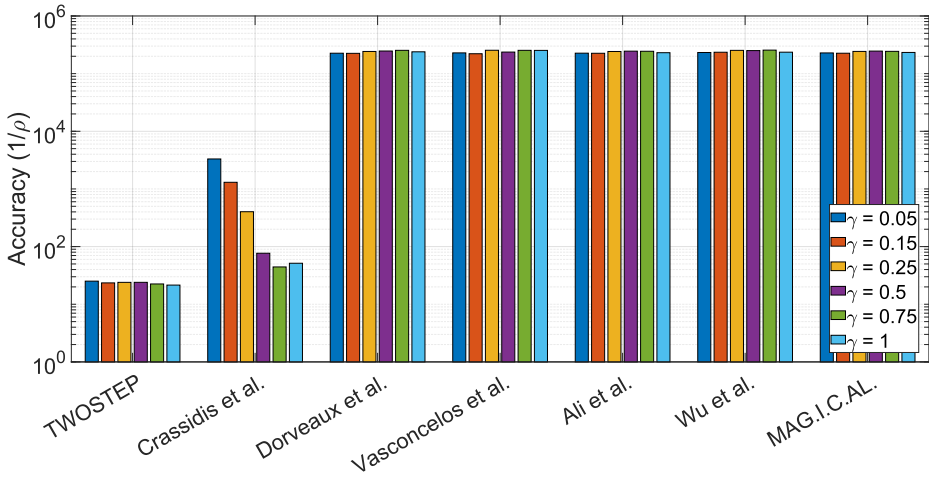
Under extreme manufacturing imperfections or the effect of hard-iron distortion, the magnitude of the offset vector, $h$, can be much larger than that in the typical case. In this Section, we examine how larger values of $\|h\|$ affect the performance of the presented algorithms. To do so, we run six Monte Carlo simulations, each one comprised of 250 different datasets generated by following Algorithm 8. The offset vector perturbation parameter $e_i$ is simulated with gradually increasing magnitude by expanding the selection horizon $\mathcal{U}[-\gamma, \gamma]$. Afterwards, its corresponding impact on each algorithm's robustness and accuracy is investigated. The distributions of the model disturbances and measurement noise are:

$$\begin{aligned} \alpha &\sim \mathcal{U}[0.8, 1.2] \\ E_{ij} &\sim \mathcal{U}[0.05, 0.05] \\ e_i &\sim \mathcal{U}[-\gamma_l, \gamma_l] \\ \sigma &= 0.005 \end{aligned}$$

for various $\gamma$

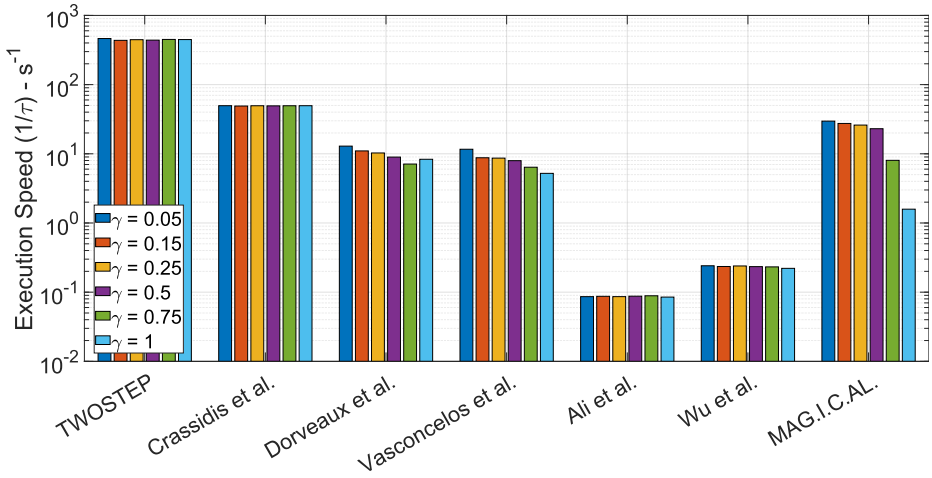
$$\gamma = \{0.05, 0.15, 0.25, 0.5, 0.75, 1\}$$

where $l = 1, 2, \ldots, 6$ is the index of Monte Carlo simulation. The extreme case of $\gamma = 1$ addresses the possibility of bias being clearly comparable and even indistinguishable to the true magnetic vector. Therefore, as $\gamma$ increases, the algorithms were driven to their limits and their functionality range was identified. All the other parameters were nominal, to ensure a fair comparison. The results of the six Monte Carlo simulations are presented in Figure 2.

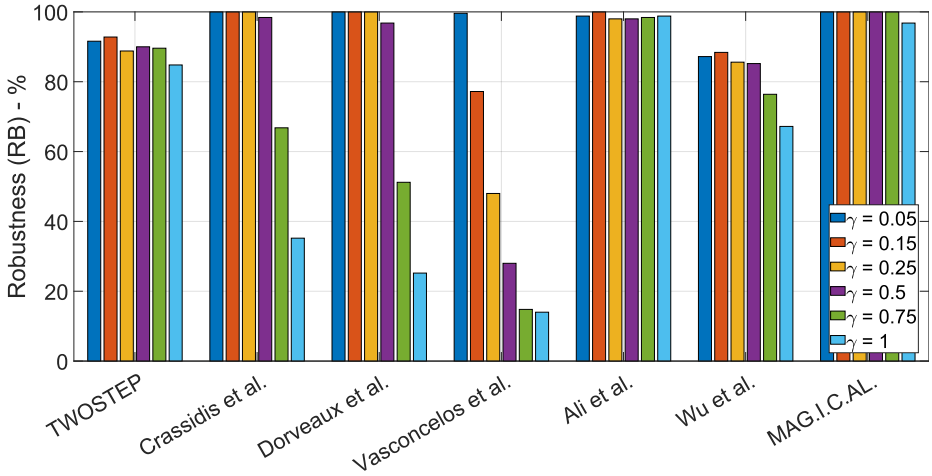



(**a**) Accuracy ($1/\rho$) of the presented algorithms for different values of $\gamma$.

**Figure 2.** *Cont.*




(**b**) Execution speed $(1/\tau)$ of the presented algorithms for different values of $\gamma$.




(**c**) Robustness $(RB(\%))$ of the presented algorithms for different values of $\gamma$.

**Figure 2.** Performance characteristics of the presented algorithms for different values of $\gamma$.

The MAG.I.C.AL and the PSO methods are the most robust ones since they function almost always, even for large values of bias, while TWOSTEP and Wu and Shi's algorithms are a little less stable. In addition, Dorveaux et al. algorithm and EKF seem to be reliable for small to moderate values of bias. All algorithms, except TWOSTEP and EKF are extremely precise when they function properly. No changes in execution speed are noticed, with the exception of MAG.I.C.AL which probably requires more iterations as the bias increases.

#### 10.1.4. The Effect of the Calibration Matrix Perturbation Parameter, $\beta$

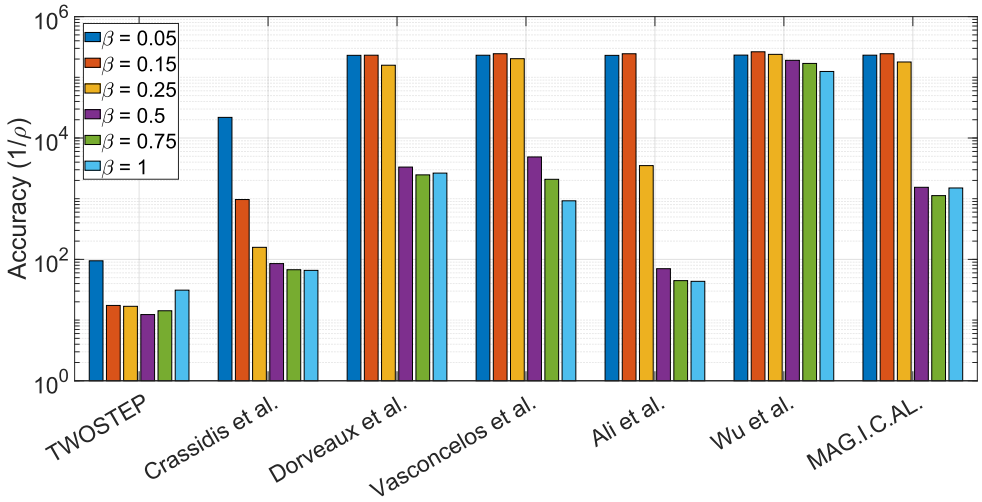
Similar to the case of the offset vector, $h$, under extreme manufacturing imperfections or the effect of soft-distortion, matrix $T$, can also diverge significantly from the typical case of the identity matrix. In this Section, we examine how larger values of perturbation $E$ affect the performance of the presented algorithms. To do so, we run six Monte Carlo simulations, each one based on 250 different datasets generated by following Algorithm 8. The perturbation elements $E_{ij}$ were simulated with gradually increasing magnitude by expanding the distribution range $\mathcal{U}[-\beta, \beta]$. Afterwards, its corresponding impact on each algorithm's robustness and accuracy is investigated. The distributions of the model disturbances and measurement noise are:

$$\alpha \sim \mathcal{U}[0.8, 1.2]$$
$$E_{ij} \sim \mathcal{U}[-\beta_l, \beta_l]$$
$$e_i \sim \mathcal{U}[-0.05, 0.05]$$
$$\sigma = 0.005$$

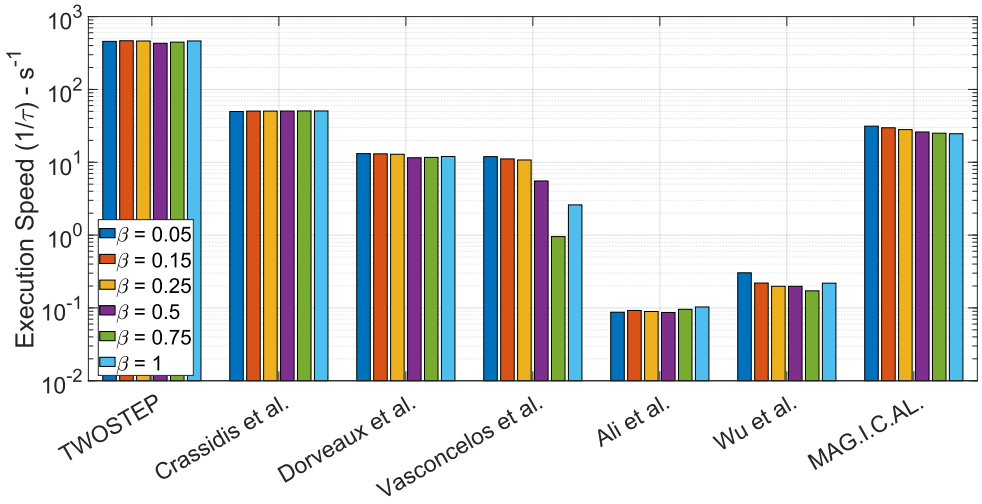
for various $\beta$

$$\beta = \{0.05, 0.15, 0.25, 0.5, 0.75, 1\}$$

where $l = 1, 2, \ldots, 6$ is the index of Monte Carlo simulation. As $\beta$ increases, the algorithms were driven to their limits and their functionality range was identified. All the other parameters were nominal, to ensure a fair comparison. The results of the six Monte Carlo simulations are presented in Figure 3.

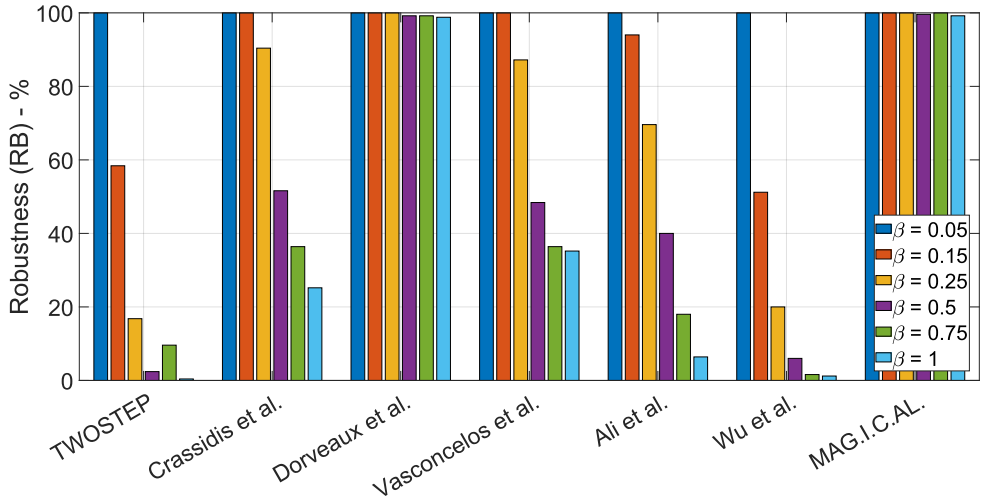



(**a**) Accuracy $(1/\rho)$ of the presented algorithms for different values of $\beta$.




(**b**) Execution Speed $(1/\tau)$ of the presented algorithms for different values of $\beta$.

**Figure 3.** *Cont.*




(**c**) Robustness ($RB(\%)$) of the presented algorithms for different values of $\beta$.

**Figure 3.** Performance characteristics of the presented algorithms for different values of $\beta$.

The MAG.I.C.AL algorithm and the algorithm of Dorveaux et al. appear to be the most robust and effective, with similar accuracy. The algorithm of Vasconcelos et al., the PSO algorithm and the EKF algorithm succeed only for small to moderate non-orthogonality errors. Vasconcelos et al. achieves accuracy comparable to that of MAG.I.C.AL. The rest of the algorithms tend to fail frequently as these errors increase. What is surprising is that Wu and Shi's algorithm provides the most accurate solutions for all $\beta$ values, but with very low robustness. To conclude, most algorithms handle bias distortion better than non-orthogonality errors.

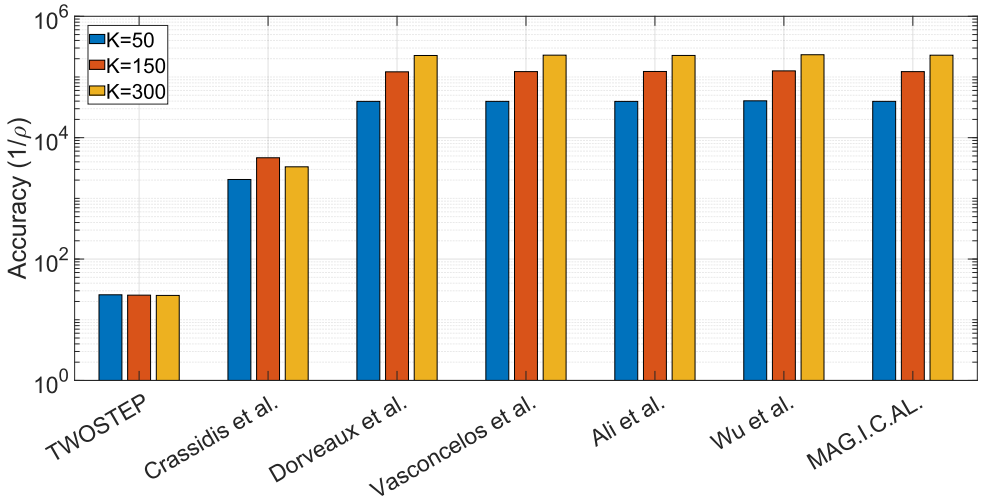
#### 10.1.5. The Effect of Dataset Size, $K$

In this section, we examine how the dataset size, $K$, affects the algorithms' performance. In general, the diversity of the measurement directions is more crucial than the quantity of them, e.g., a dataset of 50 measurements with directions distributed near uniformly on the unit sphere is significantly more suitable for the algorithms than one with thousands of measurements all having approximately the same direction.

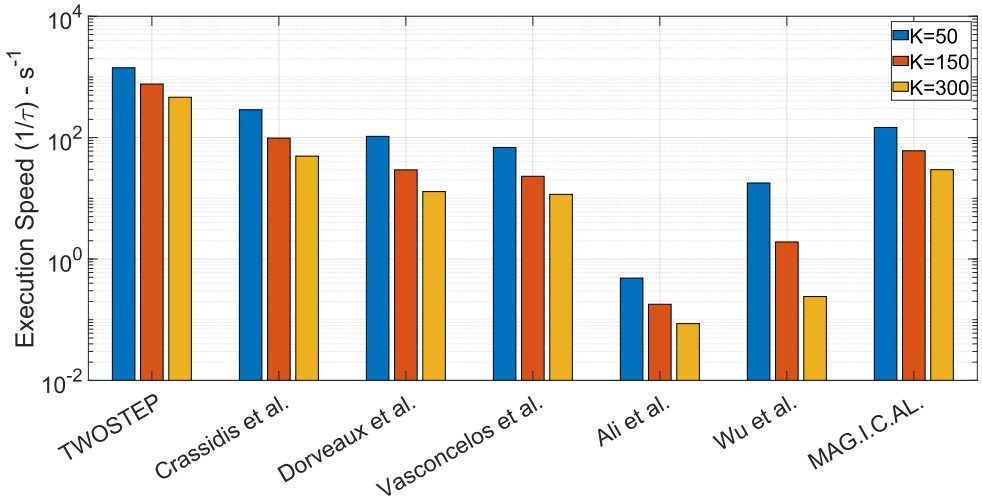
According to existing literature [4,5,7], an order of 300 measurements with directions sufficiently covering the unit sphere form an acceptable dataset for the calibration. Here we use datasets with 50, 150, and 300 measurements to test the algorithms' limits. To do so, we run three Monte Carlo simulations, based on 250 different datasets generated by Algorithm 8. The distributions of the model disturbances and measurement noise are:

$$\begin{aligned} \alpha &\sim \mathcal{U}[0.8, 1.2] \\ E_{ij} &\sim \mathcal{U}[-0.05, 0.05] \\ e_i &\sim \mathcal{U}[-0.05, 0.05] \\ \sigma &= 0.005 \end{aligned}$$

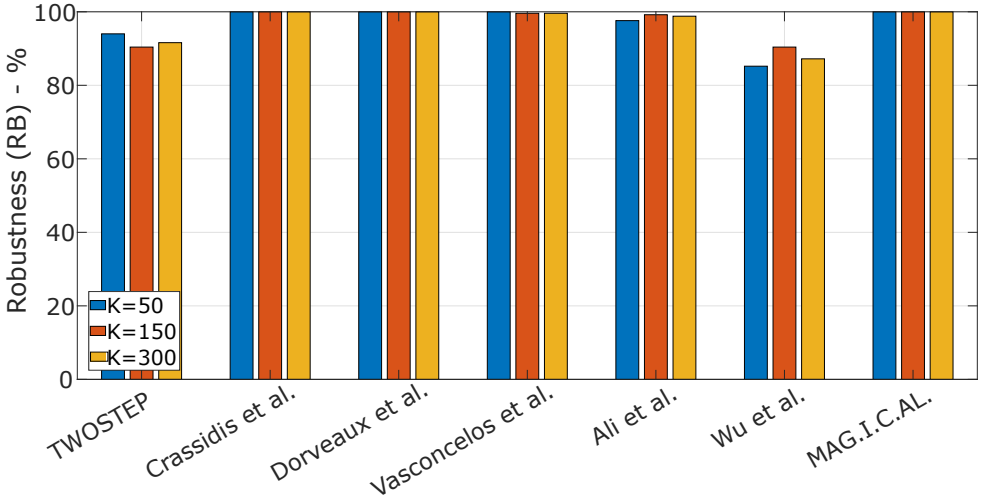
The dataset size $K$ varied whereas the distributions' ranges were fixed to nominal to ensure a fair comparison. The results of the three Monte Carlo simulations are presented in Figure 4.




(**a**) Accuracy $(1/\rho)$ of the presented algorithms for datasets of different size, $K$.




(**b**) Execution speed $(1/\tau)$ of the presented algorithms for datasets of different size, $K$.




(**c**) Robustness $(RB(\%))$ of the presented algorithms for datasets of different size, $K$.

**Figure 4.** Performance characteristics of the presented algorithms for different values of $K$.

In general, the dataset size, $K$, does not seem to be important in terms of robustness. Accuracy is surprisingly high even with only 50 measurements, which is probably

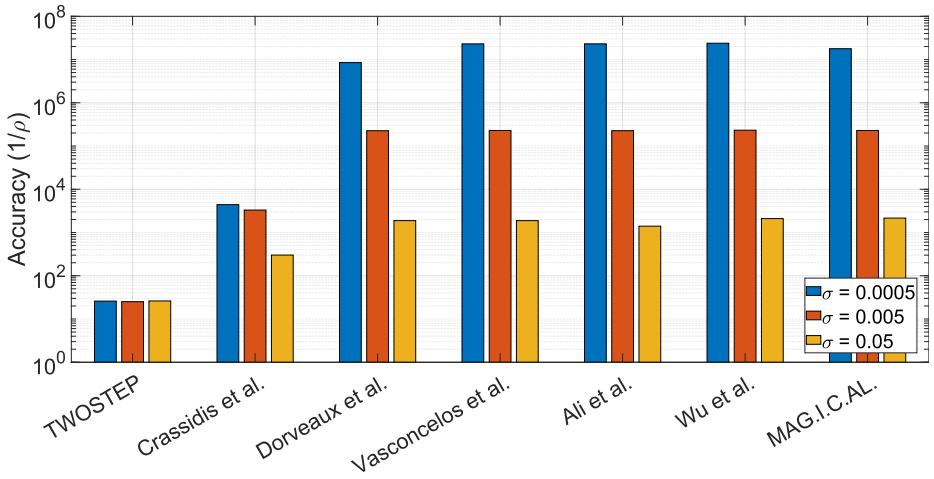
an outcome of the well distributed measurement directions using the Fibonacci lattice. Furthermore, the algorithms execution time appeared to be linear with $K$.

10.1.6. The Effect of the Noise Variance, $\sigma^2$

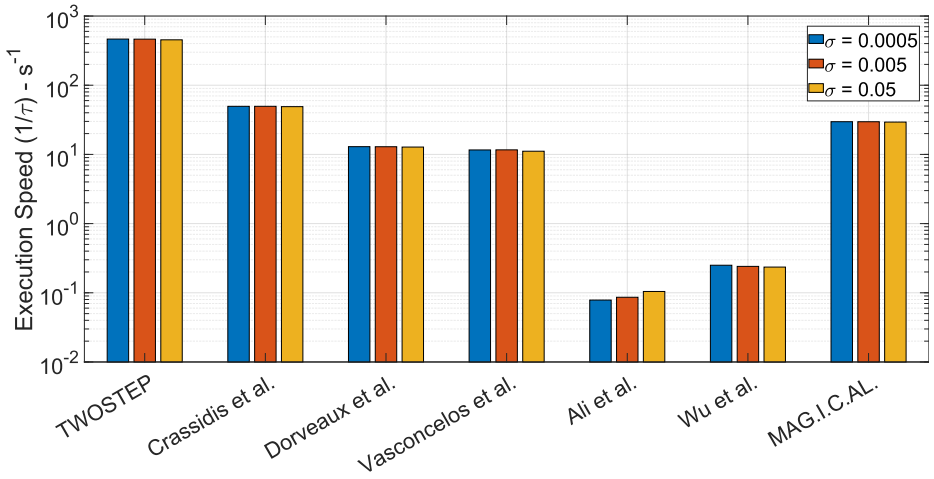
In this section, we examine the influence of measurement noise variance $\sigma$ on algorithms' robustness and accuracy. The assumption of pure white Gaussian noise in the measurement model was done. We considered a nominal value of $\sigma = 0.005$, following [2,4], while we also simulated the cases of more noisy ($\sigma = 0.05$) and less noisy ($\sigma = 0.0005$) sensors. With these choices, we represented algorithms' capabilities under 3 different orders in the magnitude of the error in the measurement. To do so, we run three Monte Carlo simulations, each one based on 250 different datasets generated by following Algorithm 8. The distributions of the model disturbances and measurement noise are:

$$\begin{aligned} \alpha &\sim \mathcal{U}[0.8, 1.2] \\ E_{ij} &\sim \mathcal{U}[-0.05, 0.05] \\ e_i &\sim \mathcal{U}[-0.05, 0.05] \\ \sigma &= \{0.0005, 0.005, 0.05\} \end{aligned}$$

Finally, all parameters except $\sigma$ were set to their default ones, to ensure a fair comparison. The results of the three Monte Carlo simulations are presented in Figure 5.

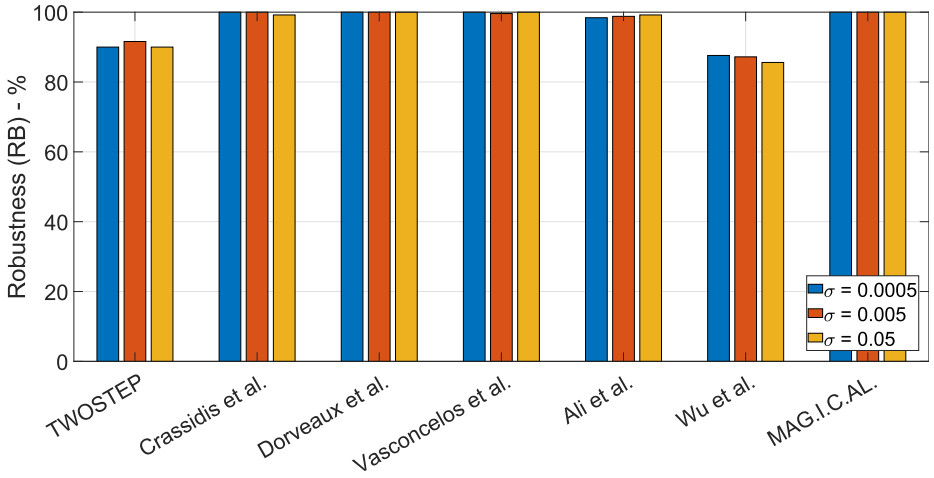



(**a**) Accuracy $(1/\rho)$ of the presented algorithms for different values of the noise variance, $\sigma^2$.




(**b**) Execution speed $(1/\tau)$ of the presented algorithms for different values of the noise variance, $\sigma^2$.

**Figure 5.** *Cont.*




(**c**) Robustness ($RB(\%)$) of the presented algorithms for different values of the noise variance, $\sigma^2$.

**Figure 5.** Performance characteristics of the presented algorithms for different values of the noise variance, $\sigma^2$.

All algorithms appear to be immune to the change of measurement's output variance $\sigma$. What is worth mentioning is that an increase of one order in variance resulted to a decrease of one order in accuracy for most algorithms (i.e., MAG.I.C.AL, Ali et al., Vasconcelos et al., Dorveaux et al., Wu and Shi).

### *10.2. Algorithm Evaluation Using Real Data*

In this section, the aforementioned algorithms are tested using real data. Multiple datasets captured by low-cost magnetic field sensors were used to verify the algorithms' performance under real-world conditions. In this case, parameters $T_{true}$ and $h_{true}$ are not known in advance. Therefore, the accuracy metric (74) cannot be used. Since, the measurements took place in a specific location, a constant magnitude of magnetic vector, $\|m\| = 1$ was considered. As a result, a proper cost function to evaluate an algorithm's effectiveness is the following

$$J_r = \frac{1}{K}\sum_{i=1}^{K}\Big(\|m_k\|^2 - 1\Big)^2 \tag{81}$$

where $K$ is the number of measurements and $k = 1, 2, \ldots, K$ is the measurement index. The estimated magnetic field vector $m_k$ for each $k$ is given by

$$m_k = T^{-1}(y_k - h) \tag{82}$$

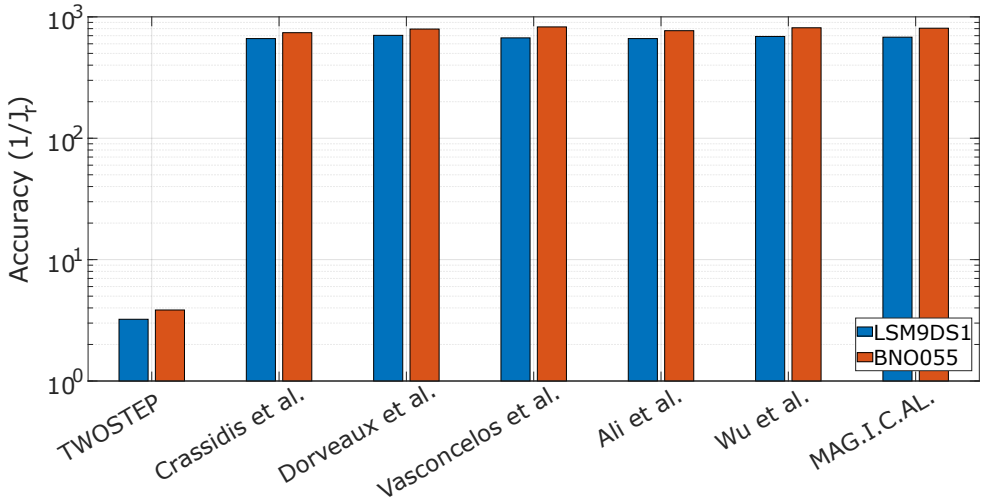
where $T$ and $h$ are the outputs of a calibration algorithm. Such a cost function is described by Wu and Shi (52), as well as by Papafotis and Sotiriadis (65).

To evaluate the performance of the presented algorithms, we used two off-the-shelf, low-cost magnetic field sensors, which are typically found in commercial electronic devices, such as smartphones, activity trackers, etc. More specifically, we captured a total of 30 datasets using the LSM9DS1 by STMicroelectronics and the BNO055 by Bosch Sensortec. The operation parameters of the two sensors during the experiment are presented in Table 3.

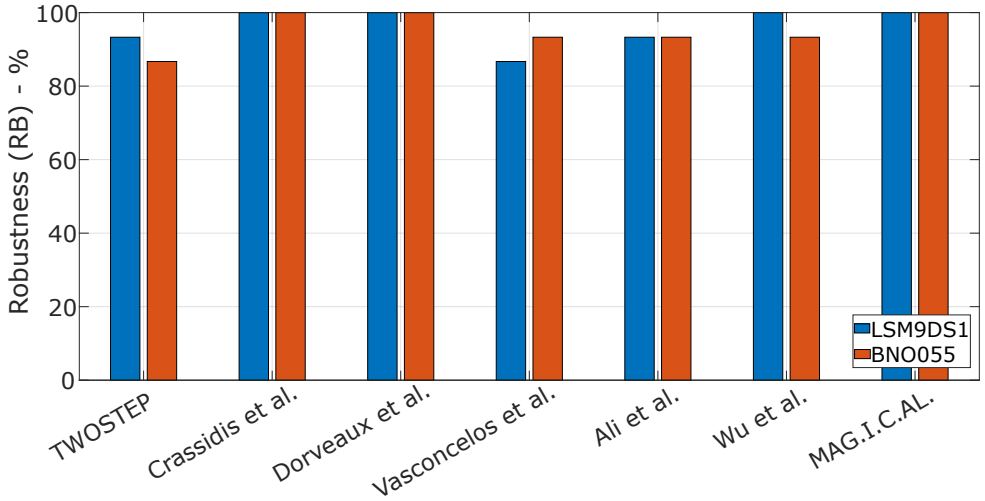
During the experiment, two sensors were fixed on the same rigid platform which was rotated by hand in several orientations. In Figure 6a, the mean value of the cost function (81) across all the recorded datasets for every algorithm is presented as a metric of accuracy. The robustness of each algorithm, as defined in (77) is presented in Figure 6b. Note that both Figure 6a,b are in agreement with the results obtained in Section 10.1.2 where synthetic data with typical values for sensor's noise and measurement distortion were considered.

**Table 3.** Operation parameters of the two magnetic field sensors.

| | **BNO055** | **LSM9DS1TR** |
|---|---|---|
| Measurement Range | ±13 Gauss | ±4 Gauss |
| Sampling Rate | 30 Hz | 80 Hz |
| Measurement Resolution | 16 bits | 16 bits |




(**a**) Accuracy $(1/J_r)$ of the presented algorithms using multiple datasets of real data from two different commercial magnetic field sensors.




(**b**) Robustness $(RB(\%))$ of the presented algorithms using multiple datasets of real data from two different commercial magnetic field sensors.

**Figure 6.** Performance characteristics of the presented algorithms using multiple datasets of real data from two different commercial magnetic field sensors.

## 11. Conclusions

To summarize, a complete and extensive study on calibration methods for low-cost magnetometers was carried out by the authors. Seven algorithms were selected for this purpose according to their popularity and their performance. A standard, unified, and complete linear measurement model was used here as the reference model for analyzing all calibration methods. After establishing the full calibration problem, these seven algorithms were discussed and were presented in an easy-to-implement way.

In order to evaluate fairly the presented algorithms' performance, we proposed a method for optimally generating artificial magnetometer data. This method was used for executing a plethora of Monte Carlo simulations. The evaluation metrics we focused on were the robustness, the accuracy and the efficiency of the algorithms. We designed several experiments to check the behavior of the algorithms under different values in bias, different values in non-orthogonality errors, different number of measurements and finally under various orders of variance in noise. Finally, several datasets of real magnetometer's data, from two different, low-cost, commercial sensors were used to verify the results obtained using the artificial data.

The following summarizes our findings regarding the studied algorithms and their possible implementation. Except from the objective criteria that we established in Section 10 to evaluate and compare the presented algorithms (accuracy, robustness, computational efficiency), in Table 4 we also evaluate the algorithms in terms of simplicity. Simplicity is used as a (subjective) metric describing our personal experience developing and testing the algorithms. It is related both to the algorithmic complexity of the algorithms (which is not an inherent disadvantage) and the quality of their presentation in the original manuscripts. The algorithms are discussed in chronological order of publication.

- ✓ **TWOSTEP**: Extremely time efficient. Works effectively for small distortions. Has low accuracy in general. The method can be generalized to on-orbit calibration.
- ✓ **Crassidis et al.**: Easy to implement. Extremely time efficient. Works effectively for small to medium distortions. The method can be generalized to on-orbit calibration. It is the only algorithm that provides online update. It can be considered as a more accurate and effective version of TWOSTEP with similar time complexity.
- ✓ **Dorveaux et al.**: Easy to implement. Moderately time efficient. Robust and accurate, but vulnerable to large values of bias.
- ✓ **Vasconcelos et al.**: Difficult to implement. Characterized by high time-complexity. Exceptional accuracy and robustness for small distortions.
- ✓ **Ali et al.**: Robust and accurate. Very high computational cost. Some prior knowledge of the search space is beneficial. At the beginning of the algorithm, the unknown variables are randomized and, thus, it is not always ensured that the algorithm will reach an optimal point. Thus, a couple of repetitions might be needed. Using modern PSO algorithms which can constrain the search space and handle a few variable inequalities increases the algorithm's performance significantly.
- ✓ **Wu and Shi**: Difficult to implement. Characterized by high time-complexity. Exceptional accuracy even with larger distortion. We noticed a 1% failure of finding an initial estimate due to inadequacy of applying Cholesky decomposition.
- ✓ **MAG.I.C.AL**: Easy to implement. Moderately time efficient. Exceptional robustness and accuracy in both synthetic and real data experiments.

**Table 4.** Algorithms' Comparison Summary – More checkmarks correspond to better performance regarding a specific metric.

| Algorithm | Simplicity | Robustness | Accuracy | Efficiency |
|---|---|---|---|---|
| TWOSTEP | ✓✓ | ✓✓ | ✓ | ✓✓✓ |
| Crassidis et al. | ✓✓✓ | ✓✓ | ✓ | ✓✓✓ |
| Dorveaux et al. | ✓✓✓ | ✓✓✓ | ✓✓✓ | ✓✓ |
| Vasconcelos et al. | ✓ | ✓ | ✓✓ | ✓ |
| Ali et al. | ✓✓ | ✓✓✓ | ✓✓✓ | ✓ |
| Wu and Shi | ✓ | ✓ | ✓✓✓ | ✓ |
| MAG.I.C.AL | ✓✓✓ | ✓✓✓ | ✓✓✓ | ✓✓ |

To conclude, in this work, we tried to cover a broad range of realistic cases and test the limits of the algorithms, noting that in real life the performance requirements differ from application to another. In some applications computational efficiency may be of major importance while great accuracy may not be needed, while in others, a very accurate calibration is essential even if significantly more computation time is required for this. Thus, there is no "perfect" algorithm appropriate for all applications; different algorithms may be more appropriate for different cases.

**Author Contributions:** Investigation, K.P., D.N. and P.P.S.; Writing—original draft, K.P., D.N.; Writing—review and editing, K.P., D.N. and P.P.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme "Human +Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A

*Appendix A.1. Gradient Vector and Hessian Matrix for the Algorithm of Vasconcelos et al.*

This section (Section 6) presents the analytical algebraic expressions for the gradient and Hessian of the likelihood function (35), used in descent optimization methods. Let $u_k = y_k - h$, the gradient of the likelihood function $J$ (33) is denoted by $\nabla J|_x = \left[\nabla J|_{\hat{T}} \nabla J|_h\right]$ and described by the submatrices

$$\nabla J|_{\hat{T}} = \sum_{k=0}^{N} \frac{2c_k}{\sigma^2} u_k \otimes \hat{T} u_k \tag{A1a}$$

$$\nabla J|_h = \sum_{k=0}^{N} \frac{-2c_k}{\sigma^2} \hat{T}^T \hat{T} u_k \tag{A1b}$$

where $c_k = 1 - \|\hat{T} u_k\|^{-1}$. The Hessian

$$\nabla^2 J|_x = \begin{bmatrix} H_{\hat{T},\hat{T}} & H_{\hat{T},h} \\ H_{\hat{T},h}^T & H_{h,h} \end{bmatrix} \tag{A2}$$

is given by the following submatrices

$$H_{\hat{T},\hat{T}} = \sum_{k=1}^{N} \frac{2}{\sigma^2} \left[ \frac{(u_k u_k^T) \otimes (\hat{T} u_k u_k^T \hat{T}^T)}{\|\hat{T} u_k\|^3} + c_k \left[ (u_k u_k^T) \otimes I_{3\times 3} \right] \right] \tag{A3a}$$

$$H_{\hat{T},h} = \sum_{k=1}^{N} \frac{-2}{\sigma^2} \left[ \frac{(u_k \otimes \hat{T} u_k) u_k^T \hat{T}^T \hat{T}}{\|\hat{T} u_k\|^3} + c_k \left[ u_k \otimes \hat{T} + I_{3\times 3} \otimes \hat{T} u_k \right] \right] \tag{A3b}$$

$$H_{h,h} = \sum_{k=1}^{N} \frac{2}{\sigma^2} \left[ \frac{\hat{T}^T \hat{T} u_k u_k^T \hat{T}^T \hat{T}}{\|\hat{T} u_k\|^3} + c_k \hat{T}^T \hat{T} \right] \tag{A3c}$$

*Appendix A.2. Gradient Vector and Hessian Matrix for the Algorithm of Wu and Shi*

This section (Section 8) presents the algebraic expressions for the gradient and Hessian of the likelihood function (51), used in descent optimization methods. For notational simplicity $\hat{T}$ and $\hat{m}$ are replaced by $T$ and $m$. Let $u_k = y_k - h$, the Jacobian vector and Hessian matrix are, respectively, derived as

$$\nabla J|_x = \begin{bmatrix} \mathbf{J}_T^T & \mathbf{J}_h^T & \underbrace{\mathbf{J}_{m_k}^T}_{k=1:N} & \underbrace{\mathbf{J}_{\lambda_k}^T}_{k=1:N} \end{bmatrix}^T \tag{A4}$$

$$\nabla^2 J|_x = \begin{bmatrix} H_{TT} & H_{Th} & H_{Tm_k} \dots & 0_{9\times 1} \dots \\ H_{Th}^T & H_{hh} & H_{hm_k} \dots & 0_{3\times 1} \dots \\ H_{Tm_k}^T & H_{hm_k}^T & H_{m_k m_k} \dots & H_{m_k \lambda_k} \dots \\ \vdots & \vdots & \vdots & \vdots \\ 0_{9\times 1}^T & 0_{3\times 1}^T & H_{m_k \lambda_k}^T \dots & 0 \dots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{A5}$$

where

$$\begin{aligned} \mathbf{J}_T &= -2 \sum_{k=1}^{N} m_k \otimes (u_k - T m_k) \\ \mathbf{J}_h &= -2 \sum_{k=1}^{N} (u_k - T m_k) \\ \mathbf{J}_{m_k} &= -2 T^T (u_k - T m_k) + 2 \lambda_k m_k \\ \mathbf{J}_{\lambda_k} &= \|m_k\|^2 - 1 \end{aligned} \tag{A6}$$

and

$$\begin{aligned} H_{TT} &= 2 \sum_{k=1}^{N} \left( m_k m_k^T \right) \otimes I \\ H_{Th} &= 2 \sum_{k=1}^{N} m_k \otimes I \\ H_{Tm_k} &= 2((m_k \otimes I) T - I \otimes (u_k - T m_k)) \\ H_{hh} &= 2NI \\ H_{hm_k} &= 2T \\ H_{m_k m_k} &= 2T^T T + 2\lambda_k I \\ H_{m_k \lambda_k} &= 2m_k \end{aligned} \tag{A7}$$

Note that in [4], the calibration matrix, $T$, is considered to be an upper triangular matrix. Thus, from both the gradient vector and the Hessian matrix, the rows and columns that correspond to the lower triangular elements of $T$ must be removed.

## References


1. Alonso, R.; Shuster, M.D. Complete Linear Attitude-Independent Magnetometer Calibration. *J. Astronaut. Sci.* **2002**, *50*, 477–490. [CrossRef]
2. Vasconcelos, J.F.; Elkaim, G.; Silvestre, C.; Oliveira, P.; Cardeira, B. Geometric Approach to Strapdown Magnetometer Calibration in Sensor Frame. *IEEE Trans. Aerosp. Electron. Syst.* **2011**, *47*, 1293–1306. [CrossRef]
3. Dorveaux, E.; Vissière, D.; Martin, A.; Petit, N. Iterative calibration method for inertial and magnetic sensors. In Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference, Shanghai, China, 15–18 December 2009; pp. 8296–8303. [CrossRef]
4. Wu, Y.; Shi, W. On Calibration of Three-Axis Magnetometer. *IEEE Sens. J.* **2015**, *15*, 6424–6431. [CrossRef]
5. Papafotis, K.; Sotiriadis, P.P. MAG.I.C.AL.—A Unified Methodology for Magnetic and Inertial Sensors Calibration and Alignment. *IEEE Sens. J.* **2019**, *19*, 8241–8251. [CrossRef]
6. Crassidis, J.; Lai, K.L.; Herman, R.R. Real-Time Attitude-Independent Three-Axis Magnetometer Calibration. *J. Guid. Control Dyn.* **2005**, *28*, 115–120. [CrossRef]
7. Ali, A.; Siddharth, S.; Syed, Z.; El-Sheimy, N. Swarm Optimization-Based Magnetometer Calibration for Personal Handheld Devices. *Sensors* **2012**, *12*, 12455–12472. [CrossRef]
8. Papafotis, K.; Sotiriadis, P.P. Accelerometer and Magnetometer Joint Calibration and Axes Alignment. *Technologies* **2020**, *8*, 11. [CrossRef]
9. Kok, M.; Hol, J.D.; Schön, T.B.; Gustafsson, F.; Luinge, H. Calibration of a magnetometer in combination with inertial sensors. In Proceedings of the 2012 15th International Conference on Information Fusion, Singapore, 9–12 July 2012; pp. 787–793.

10. Wu, Y.; Zou, D.; Liu, P.; Yu, W. Dynamic Magnetometer Calibration and Alignment to Inertial Sensors by Kalman Filtering. *IEEE Trans. Control Syst. Technol.* **2018**, *26*, 716–723. [CrossRef]
11. Kok, M.; Schön, T.B. Maximum likelihood calibration of a magnetometer using inertial sensors. *IFAC Proc. Vol.* **2014**, *47*, 92–97. [CrossRef]
12. Li, X.; Li, Z. A new calibration method for tri-axial field sensors in strap-down navigation systems. *Meas. Sci. Technol.* **2012**, *23*, 105105. [CrossRef]
13. Cao, G.; Xu, X.; Xu, D. Real-Time Calibration of Magnetometers Using the RLS/ML Algorithm. *Sensors* **2020**, *20*, 535. [CrossRef] [PubMed]
14. Hadjigeorgiou, N.; Asimakopoulos, K.; Papafotis, K.; Sotiriadis, P.P. Vector Magnetic Field Sensors: Operating Principles, Calibration and Applications. *IEEE Sens. J.* **2020**, 21, 12531–12544. [CrossRef]
15. IAGA Division V; Working Group 8. Revision of International Geomagnetic Reference Field released. *EOS Trans.* **1996**, *77*, 153. [CrossRef]
16. Gambhir, B. *Determination of Magnetometer Biases Using Module RESIDG*; Technical Report; Computer Sciences Corporation: Falls Church, VA, USA, 1975.
17. LERNER, G. *Spacecraft Attitude Determination and Control*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1978
18. Alonso, R.; Shuster, M.D. TWOSTEP: A fast robust algorithm for attitude-independent magnetometer-bias determination. *J. Astronaut. Sci.* **2002**, *50*, 433–451. [CrossRef]
19. Strang, G. *Linear Algebra and Its Applications*; Brooks Cole/Cengage Learning: Belmont, CA, USA 2007
20. Crassidis, J.L.; Markley, F.L.; Lightsey, E.G. Global Positioning System Integer Ambiguity Resolution without Attitude Knowledge. *J. Guid. Control Dyn.* **1999**, *22*, 212–218. [CrossRef]
21. Crassidis, J.L. *Optimal Estimation of Dynamic Systems*; CRC Press: Boca Raton, FL, USA, 2004.
22. Springmann, J.C.; Cutler, J.W. Attitude-Independent Magnetometer Calibration with Time-Varying Bias. *J. Guid. Control Dyn.* **2012**, *35*, 1080–1088. [CrossRef]
23. Foster, C.C. Elkaim, Extension of a two-step calibration methodology to include nonorthogonal sensor axes. *IEEE Trans. Aerosp. Electron. Syst.* **2008**, *44*, 1070–1078. [CrossRef]
24. Gebre-Egziabher, D.; Elkaim, G.; Powell, J.; Parkinson, B. Calibration of Strapdown Magnetometers in Magnetic Field Domain. *J. Aerosp. Eng.* **2006**, *19*, 87–102. [CrossRef]
25. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
26. Kennedy, J.; Obaiahnahatti, B.; Shi, Y. *Swarm Intelligence*; Morgan Kaufmann Academic Press: San Francisco, CA, USA, 2001; Volume 1, pp. 1931–1938.
27. Magnus Erik Hvass Pedersen. Good Parameters for Particle Swarm Optimization; Hvass Laboratories. 2010. Available online: https://www.semanticscholar.org/paper/Good-Parameters-for-Particle-Swarm-Optimization-Pedersen/a4ad7500b64d70a2ec84bf57cfc2fedfdf770433 (accessed on 31 July 2021)
28. Mezura-Montes, E.; Coello, C. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm Evol. Comput.* **2011**, *1*, 173–194. [CrossRef]
29. Helwig, S.; Branke, J.; Mostaghim, S. Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization. *IEEE Trans. Evol. Comput.* **2013**, *17*, 259–271. [CrossRef]
30. MATLAB. *Optimization Toolbox*; The MathWorks Inc.: Natick, MA, USA, 1999.
31. Roberts, M. How to Evenly Distribute Points on a Sphere More Effectively than the Canonical Fibonacci Lattice. Available online: http://extremelearning.com.au/how-to-evenly-distribute-points-on-a-sphere-more-effectively-than-the-canonical-fibonacci-lattice/ (accessed on 22 May 2021).
32. Gonzalez, A. Measurement of Areas on a Sphere Using Fibonacci and Latitude–Longitude Lattices. *Math. Geosci.* **2010**, *42*, 49–64. [CrossRef]
33. Schonemann, P. A generalized solution of the orthogonal procrustes problem. *Psychometrika* **1966**, *31*, 1–10. [CrossRef]