Robust Autocalibration of Triaxial Magnetometers

Je Hyeong Hong, *Member, IEEE*, Donghoon Kang, *Member, IEEE*, and Ig-Jae Kim, *Member, IEEE*

***Abstract*— Self-calibration of a magnetometer usually requires controlled magnetic environment as the calibration output can be affected by field distortions from nearby magnetic objects. In this article, we develop a two-stage method that can accurately self-calibrate magnetometer from measurements containing anomalous readings due to local magnetic disturbances. The method proceeds by robustly fitting an ellipsoid to measurement data via L_1-norm convex optimization, yielding initial model variables that are less prone to magnetic disruptions. These are then served as a starting point for robust nonlinear least-squares optimization, which refines the magnetometer model to minimize sensor estimation errors while suppressing heavy anomalies. Synthetic and real experimental results are provided to demonstrate improved accuracy of the proposed method in the presence of outliers. We additionally show empirically that the method is directly applicable to self-calibration of three-axis accelerometers.**

***Index Terms*— Accelerometer, calibration, magnetometer, nonlinear least squares, robust optimization.**

I. Introduction

MAGNETOMETER is a widely used sensor across a range of tasks, including attitude control [1], [2], navigation and mapping [3]–[5], and metal object detection [6]. It is a constituent of an inertial measurement unit (IMU), which is a key component for localization tasks.

A majority of recently manufactured magnetometers comprises three axes each of which detects the Hall voltage that gives an indication of the magnetic field direction. The relationship between the output reading and the input field is affected by several factors, including soft and hard iron effects and electrical sensor scaling and offset. While the readers are encouraged to refer to the work of Vasconcelos *et al.* [1] or Papafotis and Sotiriadis [7] for detailed descriptions, we utilize the derivation from their work and others [8]–[11] that the combined effect of these factors yields the following linear system equation for a three-axis magnetometer:

$$\mathbf{m}_j = \mathbf{y}_j + \boldsymbol{\varepsilon}_j = \mathtt{A}\hat{\mathbf{x}}_j + \mathbf{b} + \boldsymbol{\varepsilon}_j \tag{1}$$

where $\mathbf{m}_j \in \mathbb{R}^3$ is the measurement at time instant j, $\mathbf{y}_j \in \mathbb{R}^3$ is the ideal noise-free model output at time j, $\hat{\mathbf{x}}_j \in S^2$ is the corresponding true magnetic field direction vector at time step j, $\mathtt{A} \in \mathbb{R}^{3\times 3}$ is the overall sensor transformation matrix, $\mathbf{b} \in \mathbb{R}^3$ is the combined model offset, and $\boldsymbol{\varepsilon}_j \in \mathbb{R}^3$ is the measurement noise.

An autocalibration procedure typically refers to estimating $\mathtt{A}$ and $\mathbf{b}$ of the sensor that best explains the measurements $\{\mathbf{m}_j\}$ obtained from arbitrary movements of the sensor. A classic choice for such calibration has been ellipsoid fitting [12]—by noting that each field direction $\hat{\mathbf{x}}_j$ is ideally a unit vector, each ideal model output $\mathbf{y}_j$ must lie on a 3-D ellipsoid. Several works [1], [8]–[11] have utilized this constraint to estimate model variables by least-squares fitting an ellipsoid on the measurement data.

The above approach faces two drawbacks—first, as reported by Vasconcelos *et al.* [1], the minimized objective during ellipsoid fitting is an algebraic error that is suboptimal when considering the actual sensor estimation error $\|\mathbf{y}_j - \mathbf{m}_j\|$. Second, the least-squares nature of the employed ellipsoid fitting algorithm makes it vulnerable to anomalous measurements arising from local magnetic disturbances due to nearby ferromagnetic materials that do not move with the sensor.

While attempts have been made to mitigate the suboptimality issue of the ellipsoid fitting solution (e.g., by proposing to minimize the sensor estimation error in [7]), less effort has been made in improving robustness to measurement anomalies. As shown in Fig. 1, a few anomalies can deteriorate the calibration accuracy, and it is onerous to simply remove these through enforcing a more controlled calibration setting.

To address and resolve the aforementioned issues, we propose an autocalibration method, which finds an optimal model minimizing the sensor estimation errors while maintaining robustness to outlier data. The method comprises two stages, robust L_1-norm ellipsoid fitting on measurements for initial sensor model estimation followed by the refinement of model variables via robust nonlinear least-squares optimization. Through the consideration of outliers, we will show stable calibration accuracy and precision can be achieved.

The main contributions of this article are as follows:

- development of a robust two-step autocalibration algorithm for three-axis magnetometers, which yields state-of-the-art calibration performance on simulation and real data and extensive comparison of our method against baseline and state-of-the-art algorithms.
- In addition, we demonstrate empirically that our method is directly applicable to self-calibrating three-axis accelerometers.

The rest of this article is structured as follows. Section I-A reviews previous work in autocalibration of triaxial magne-

Manuscript received July 16, 2020; revised September 20, 2020; accepted October 9, 2020. Date of publication November 2, 2020; date of current version December 18, 2020. This work was supported in part by the Korea Institute of Science and Technology (KIST) Institutional Program under Project 2E30270 and in part by the Energy Technology Development Business Program of the KETEP under Grant 20181110100420. The Associate Editor coordinating the review process was Shisong Li. *(Corresponding author: Ig-Jae Kim.)*

The authors are with the Center for Artificial Intelligence, Korea Institute of Science and Technology, Seoul 02792, South Korea (e-mail: jhh37@outlook.com; kimbab.moowoo@gmail.com; drjay@kist.re.kr).

Digital Object Identifier 10.1109/TIM.2020.3035184

TABLE I

SUMMARY OF DIFFERENT ALGORITHMS FOR SELF-CALIBRATING TRIAXIAL MAGNETOMETERS. ($\mathtt{A}$, $\mathbf{b}$) ARE THE SENSOR MODEL VARIABLES, $\hat{\mathbf{x}}_j$ IS THE MAGNETIC FIELD DIRECTION AT TIME j, AND $\mathbf{m}_j$ IS THE ACTUAL MEASUREMENT AT THE CORRESPONDING TIME

Authors	Initialization (1st stage)	Optimization / refinement (2nd stage)	Robust to outliers
Foster and Elkaim [13] (2008)	L_2-norm ellipsoid fitting (least squares)	No additional refinement performed	No
Dorveaux et al. [8] (2009)	$\hat{\mathbf{x}}_j \leftarrow \mathbf{m}_j$ for each time step j	* Geometric ellipsoid fitting via alternation 1. Minimize $\sum_{j=1}^{N} \left\|\mathtt{L}\hat{\mathbf{x}}_j + \mathbf{c} - \hat{\mathbf{x}}_j/\|\hat{\mathbf{x}}_j\|_2\right\|_2^2$ over $\mathtt{L}$ and $\mathbf{c}$ 2. Update each field direction $\hat{\mathbf{x}}_j$ as $\hat{\mathbf{x}}_j \leftarrow \mathtt{L}\hat{\mathbf{x}}_j + \mathbf{c}$ 3. Update the model variables as $\mathtt{A} \leftarrow \mathtt{AL}^{-1}$, $\mathbf{b} \leftarrow \mathbf{b} - \mathtt{A}\mathbf{c}$ 4. Repeat steps 1–3 until convergence.	No
Vasconcelos et al. [1] (2011)	L_2-norm ellipsoid fitting (least squares)	* Geometric ellipsoid fitting via nonlinear joint optimization - Minimize $\sum_{j=1}^{N}(\|\mathtt{L}(\mathbf{m}_j - \mathbf{b})\|_2 - 1)^2$ jointly over $\mathtt{L}$ and $\mathbf{b}$ and get $\mathtt{A} := \mathtt{L}^{-1}$.	No
Kok and Schön [11] (2016)	L_2-norm ellipsoid fitting (semidefinite programming)	No additional refinement performed	No
Papafotis and Sotiriadis [7] (MAGICAL, 2019)	$\hat{\mathbf{x}}_j \leftarrow \frac{\mathbf{m}_j}{\|\mathbf{m}_j\|_2}$ for each time step j	* Minimization of sensor estimation errors via alternation 1. Minimize $\sum_{j=1}^{N} \|\mathtt{A}\hat{\mathbf{x}}_j + \mathbf{b} - \mathbf{m}_j\|_2^2$ over $(\mathtt{A}, \mathbf{b})$ 2. Minimize $\sum_{j=1}^{N} \|\mathtt{A}\hat{\mathbf{x}}_j + \mathbf{b} - \mathbf{m}_j\|_2^2$ over $\{\hat{\mathbf{x}}_j\}$ 3. Normalize each $\hat{\mathbf{x}}_j$ as $\hat{\mathbf{x}}_j \leftarrow \hat{\mathbf{x}}_j / \|\hat{\mathbf{x}}_j\|_2$ 4. Repeat steps 1–3 until convergence	No
Ours (Autocal)	L_1-norm ellipsoid fitting (semidefinite programming)	* Robust minimization of sensor estimation errors via nonlinear joint optimization - Minimize $\sum_{j=1}^{N} \rho(\|\mathtt{A}\hat{\mathbf{x}}_j + \mathbf{b} - \mathbf{m}_j\|_2^2)$ jointly over $\mathtt{A}$, $\mathbf{b}$ and $\{\hat{\mathbf{x}}_j\}$ subject to $\|\hat{\mathbf{x}}_j\|_2 = 1 \ \forall\ j$	**Yes**

tometers, and Section II illustrates our proposed method in detail. Section III presents our procedures and discusses the obtained results for both simulation and real experiments. Conclusions are drawn in Section IV.

A. Related Work

As briefly stated in Section I, the most well-known self-calibration algorithm for a three-axis magnetometer is the least-squares ellipsoid fitting method [13], [14]. This method uses the fact that the magnitude of each magnetic field remains constant, leading to the sensor estimates ideally lying on a 3-D ellipsoid. Since geometric fitting of ellipsoid involves nonlinear optimization [1], this method instead minimizes an algebraic replacement of the original fitting objective that can be solved through a single step of least squares. The method benefits from easy implementation and fast computation, but the obtained algebraic solution is suboptimal with respect to the original geometric objective.

To alleviate the abovementioned issue, numerous works have focused on improving the solution quality (hence calibration accuracy). Kok and Schön [11] adopted semidefinite programming (SDP) to obtain (with guarantee) the global minimum of the algebraic cost, but this is still suboptimal with respect to the geometric fitting error. Dorveaux *et al.* [8] proposed to alternatively update the sensor variables and the magnetic field directions every iteration to make the norm of each field direction close to 1. The algorithm is efficient as it solves a sequence of linear equations, but due to the way that the sensor model is effectively initialized as $\mathtt{A} = \mathtt{I}$ (identity matrix) and $\mathbf{b} = \mathbf{0}$ (zero vector) for iterative optimization, the algorithm can yield suboptimal results (as will be shown in Section III) if the degrees of soft and hard iron effects are significant. Vasconcelos *et al.* [1] devised a two-step method, whereby an initial sensor model is obtained through least-squares ellipsoid fitting after which is refined via nonlinear optimization to make the norm of each field direction close to 1. This method makes no initial assumption but does not consider potential outlier measurements.

More recently, Papafotis and Sotiriadis [7] proposed to minimize the sum of sensor estimation errors instead of geometric ellipsoid fitting errors. This method alternatively updates sensor model variables and 3-D magnetic field directions, assuming that each field direction is of unit norm. As will be demonstrated in Section III, their objective makes a further improvement to the previous work minimizing ellipsoid fitting errors. Nevertheless, this algorithm also employs the same restrictive initialization of the sensor model as in [8] despite its local convergence, and it lacks consideration of anomalies, which can potentially deteriorate calibration accuracy.

A summary of previous works can be found in Table I.

II. PROPOSED METHOD

In this section, we illustrate our two-stage method for robust autocalibration of a three-axis magnetometer. The algorithm has been designed to consider anomalous sensor readings without requiring additional information, such as initial model

estimates. One may find Table II useful for looking up utilized symbols throughout this section.

A. Canonical Form of the Solution

As noted by Wu and Shi [10], for an optimal model $(\mathtt{A}, \mathbf{b})$ and the corresponding field directions $\{\hat{\mathbf{x}}_j\}$, any orthogonal matrix $\mathtt{Q}$ can yield an equivalent model $(\mathtt{A}\mathtt{Q}^\top, \mathbf{b})$ with field directions $\{\mathtt{Q}\hat{\mathbf{x}}_j\}$ since $\mathtt{A}\hat{\mathbf{x}}_j = (\mathtt{A}\mathtt{Q}^\top)(\mathtt{Q}\hat{\mathbf{x}}_j)$. To resolve this solution ambiguity, we adopt a similar approach to [8] and [10], forcing $\mathtt{A}$ to be an upper triangular matrix $\mathtt{K}$ during optimization. Furthermore, we additionally force the diagonal entries to be positive when outputting the model variables.

To see how above removes solution ambiguity, first, note that the RQ decomposition $\mathtt{A} = \mathtt{K}\mathtt{Q}$ (where $\mathtt{K}$ is an upper triangular matrix and $\mathtt{Q}$ is an orthogonal matrix) is unique so long as each diagonal entry of $\mathtt{K}$ is set positive [15]. By forcing $\mathtt{A}$ to be upper triangular with positive diagonal entries, the entire solution space $\{\mathtt{A}\mathtt{Q}^\top\}$ spanned by an orthogonal matrix $\mathtt{Q}$ (i.e., the Grassmann manifold [16]) is uniquely turned into $\mathtt{K}$.

B. Data Normalization

For numerical stability of our algorithm, we have applied standard data normalization techniques from [17]—i.e., each measurement is subtracted by the median of the measurement set $\{\mathbf{m}_j\}$ and then divided by its standard deviation prior to feeding through the algorithm. The solution obtained from the algorithm is then reverse-normalized.

C. Robust L_1-Norm Ellipsoid Fitting

In this section, we introduce and apply Calafiore's L_1-norm ellipsoid fitting algorithm [12] to obtain an initial sensor model robust to outliers.

As illustrated in the literature, the fact that each field direction $\hat{\mathbf{x}}_j$ lies on a unit 3-D sphere constrains each model output $\mathbf{y}_j$ to lie on a 3-D ellipsoid that is

$$\|\hat{\mathbf{x}}_j\|_2^2 = \|\mathtt{K}^{-1}(\mathbf{y}_j - \mathbf{b})\|_2^2 = 1 \tag{2}$$

leading to

$$\mathbf{y}_j^\top \mathtt{C}\mathbf{y}_j + 2\mathbf{d}^\top \mathbf{y}_j + e = 0 \tag{3}$$

where $\mathtt{C}$, $\mathbf{d}$, and e are defined up to scale as $\mathtt{C} \simeq \mathtt{K}^{-\top}\mathtt{K}^{-1}$, $\mathbf{d} \simeq -\mathtt{K}^{-\top}\mathtt{K}^{-1}\mathbf{b}$, and $e \simeq \mathbf{b}^\top\mathtt{K}^{-\top}\mathtt{K}^{-1}\mathbf{b} - 1$. By introducing a scale factor μ, these can be written as the following equations:

$$\begin{aligned} \mathtt{C} &:= \mu\mathtt{K}^{-\top}\mathtt{K}^{-1} \\ \mathbf{d} &:= -\mu\mathtt{K}^{-\top}\mathtt{K}^{-1}\mathbf{b} = -\mathtt{C}\mathbf{b} \\ e &:= \mu(\mathbf{b}^\top\mathtt{K}^{-\top}\mathtt{K}^{-1}\mathbf{b} - 1) = -\mathbf{b}^\top\mathbf{d} - \mu. \end{aligned} \tag{4}$$

In an ideal noiseless environment, each $\mathbf{m}_j = \mathbf{y}_j$ yields one unique ellipsoid equation (3), and therefore, one can solve exactly for the model variables given nine observations or more. In practice, however, measurement errors plague such exact estimation, and one needs to account for these errors.

As reviewed by Calafiore [12], minimizing the geometric error $\|\mathbf{y}_j - \mathbf{m}_j\|_2^2$ between each measurement and the closest point on the ellipsoid requires an iterative solver with good initialization. Instead, a widely used alternative is to minimize an algebraic fitting error

$$\begin{aligned} g_j(\mathtt{C}, \mathbf{d}, e) &:= \mathbf{m}_j^\top \mathtt{C}\mathbf{m}_j + 2\mathbf{d}^\top\mathbf{m}_j + e \\ &= \begin{bmatrix}\mathbf{m}_j \\ 1\end{bmatrix}^\top \begin{bmatrix}\mathtt{C} & \mathbf{d} \\ \mathbf{d}^\top & e\end{bmatrix}\begin{bmatrix}\mathbf{m}_j \\ 1\end{bmatrix} \end{aligned} \tag{5}$$

subject to $\mathtt{C} \succ 0$ for valid formation of an ellipsoid. Minimizing $\sum_{j=1}^N \|g_j\|_2^2$ yields the L_2-norm least-squares solution without need for initial variables, but due to its nature of squaring error distances, it is susceptible to anomalous data, as shown in Fig. 1(a).

TABLE II
LIST OF MATHEMATICAL SYMBOLS USED THROUGHOUT THIS ARTICLE

Symbol	Dimension	Definition
N	$\mathbb{R}$	number of measurements in a sequence
η	$\mathbb{R}$	proportion of anomaly measurements
σ	$\mathbb{R}$	standard deviation of Gaussian noise
ζ	$\mathbb{R}$	maximum error amplitude of anomalies
λ	$\mathbb{R}$	Damping factor
$\mathtt{A}$	$\mathbb{R}^{3\times3}$	sensor transformation matrix
$\mathtt{K}$	$\mathbb{R}^{3\times3}$	upper triangular form of $\mathtt{A}$
$\mathbf{b}$	$\mathbb{R}^3$	sensor offset
$\hat{\mathbf{x}}_j$	$\mathbb{R}^3$	unit magnetic field direction at time j
$\mathbf{y}_j$	$\mathbb{R}^3$	noise-free model estimate at time j
$\mathbf{m}_j$	$\mathbb{R}^3$	measurement at time j
$\boldsymbol{\varepsilon}_j$	$\mathbb{R}^3$	random noise at time j
$\boldsymbol{\xi}_j$	$\mathbb{R}^3$	anomaly-inducing error at time j
$\mathbf{m}_j$	$\mathbb{R}^3$	measurement at time j
$\hat{\mathbf{x}}$	$\mathbb{R}^{3N}$	a collection of field vectors $\mathrm{vec}([\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_N])$
$\mathbf{k}$	$\mathbb{R}^6$	elements of $\mathtt{K}$ defined as $[k_{1,1}, k_{1,2}, k_{1,3}, k_{2,2}, k_{2,3}, k_{3,3}]^\top$
$\mathtt{\Pi}$	$\mathbb{R}^{9\times6}$	projection matrix defined such that $\mathtt{\Pi}\mathbf{k} = \mathrm{vec}(\mathtt{K})$
$\mathbf{p}$	$\mathbb{R}^9$	a vector of sensor variables defined as $\mathbf{p} := [\mathbf{k}^\top, \mathbf{b}^\top]^\top$
$\boldsymbol{\theta}$	$\mathbb{R}^{9+3N}$	a collection of all variables defined as $\boldsymbol{\theta} := [\mathbf{p}^\top, \hat{\mathbf{x}}_1^\top, \cdots, \hat{\mathbf{x}}_N^\top]^\top$
$\mathbf{r}_j$	$\mathbb{R}^3$	residual at time j $(\mathbf{y}_j - \mathbf{m}_j)$
ρ	$\mathbb{R} \to \mathbb{R}$	robust kernel (see Sec. II-D2 for details)
f	$\mathbb{R}$	cost function $\sum_{j=1}^N \rho(\|\mathbf{r}_j\|_2^2)$
ρ'	$\mathbb{R} \to \mathbb{R}$	derivative of ρ
ρ''	$\mathbb{R} \to \mathbb{R}$	derivative of ρ'
$\mathtt{J}_j$	$\mathbb{R}^{3\times9+3N}$	Jacobian of the residual $\mathbf{r}_j$ w.r.t. $\boldsymbol{\theta}$
α_j	$\mathbb{R}$	Triggs-correction factor in (24)
$\tilde{\mathbf{r}}_j$	$\mathbb{R}^3$	Triggs-correction of the residual $\mathbf{r}_j$
$\tilde{\mathtt{J}}_j$	$\mathbb{R}^{3\times9+3N}$	Triggs-corrected Jacobian w.r.t. $\boldsymbol{\theta}$
$\tilde{\mathtt{J}}_{j,\mathbf{p}}$	$\mathbb{R}^{3\times9}$	Triggs-corrected Jacobian w.r.t. $\mathbf{p}$
$\tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_k}$	$\mathbb{R}^{3\times3}$	Triggs-corrected Jacobian w.r.t. $\hat{\mathbf{x}}_k$
$\mathbf{g}$	$\mathbb{R}^{9+3N}$	gradient of the function (8) w.r.t. $\boldsymbol{\theta}$
$\mathbf{g}_\mathbf{p}$	$\mathbb{R}^9$	gradient of (8) w.r.t. $\mathbf{p}$
$\mathbf{g}_{\hat{\mathbf{x}}_j}$	$\mathbb{R}^3$	gradient of (8) w.r.t. $\hat{\mathbf{x}}_j$
$\mathbf{g}_{\hat{\mathbf{x}}}$	$\mathbb{R}^{3N}$	gradient of (8) w.r.t. $\hat{\mathbf{x}}$
$\mathtt{H}$	$\mathbb{R}^{9+3N\times9+3N}$	Approx. Hessian of (8) w.r.t. $\boldsymbol{\theta}$
$\mathtt{H}_{\mathbf{p},\mathbf{p}}$	$\mathbb{R}^{9\times9}$	Approx. Hessian of (8) w.r.t. $\mathbf{p}$
$\mathtt{H}_{\hat{\mathbf{x}}_j,\hat{\mathbf{x}}_j}$	$\mathbb{R}^{3\times3}$	Approx. Hessian of (8) w.r.t. $\hat{\mathbf{x}}_j$
$\mathtt{H}_{\hat{\mathbf{x}},\hat{\mathbf{x}}}$	$\mathbb{R}^{3N\times3N}$	Approx. Hessian of (8) w.r.t. $\hat{\mathbf{x}}$
$\mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_j}$	$\mathbb{R}^{9\times3}$	$2\tilde{\mathtt{J}}_{j,\mathbf{p}}^\top\tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j}$
$\mathtt{D}_j$	$\mathbb{R}^{3\times3}$	manifold constraint term defined in (30)
$\mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}}$	$\mathbb{R}^{9\times3N}$	Off-diagonal $[\mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_1}, \cdots, \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_N}]$
$\tilde{\mathtt{H}}$	$\mathbb{R}^{9+3N\times9+3N}$	Damped approx. Hessian w.r.t. $\boldsymbol{\theta}$
$\boldsymbol{\Delta}\mathbf{p}$	$\mathbb{R}^9$	update in $\mathbf{p} := [\mathbf{k}; \mathbf{b}]$
$\boldsymbol{\Delta}\boldsymbol{\theta}$	$\mathbb{R}^{9+3N}$	update in $\boldsymbol{\theta}$

Algorithm 1 L_1-Norm Ellipsoid Fitting Algorithm for Estimation of Sensor Variables ($\mathtt{A}$, $\mathbf{b}$) and Magnetic Field Directions $\{\hat{\mathbf{x}}_j\}$

Inputs: measurements $\{\mathbf{m}_j\}$

Solve the semidefinite program (6) proposed by Calafiore [12] using SeDuMi [18] and obtain $\mathtt{C}$, $\mathbf{d}$, e.

$\mathbf{b} \leftarrow -\mathtt{C}^{-1}\mathbf{d}$

$\mu \leftarrow -\mathbf{b}^\top\mathbf{d} - e$

$\mathtt{K} \leftarrow \sqrt{\mu}\ \mathrm{chol}(\mathtt{C})^{-1}$

for $j = 1, \ldots, N$ **do**

  $\hat{\mathbf{x}}_j \leftarrow \mathtt{K}^{-1}(\mathbf{m}_j - \mathbf{b})$

  $\hat{\mathbf{x}}_j \leftarrow \hat{\mathbf{x}}_j / \|\hat{\mathbf{x}}_j\|_2$

end for

Outputs: $\mathtt{K}$, $\mathtt{b}$, $\{\hat{\mathbf{x}}_k\}$

In order to gain robustness to outliers, Calafiore [12] proposed to minimize $\sum_{j=1}^{N}\|g_j\|_1$ by formulating it as a form of SDP [19], whose global minimum can be found using a convex optimization algorithm. To briefly rederive the SDP form, minimizing $\sum_{j=1}^{N}\|g_j\|_1$ is equivalent to minimizing $\sum_{j=1}^{N}\gamma_j$ subject to $\gamma_j > \|g_j\|_1 \forall\, j$. Note that this inequality constraint can be replaced by a pair of constraints $\gamma_j > g_j$ and $\gamma_j > -g_j$. Finally, adding the constraint $\mathrm{trace}(\mathtt{C}) = 1$ for enforcing $\mathtt{C} \succ 0$ (positive definiteness for an ellipsoid) yields a semidefinite program

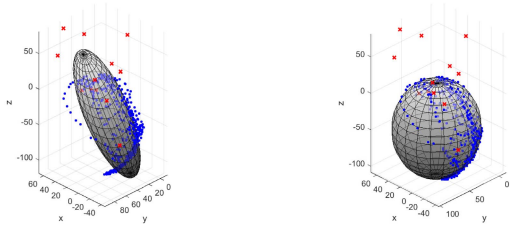
$$\begin{aligned} \underset{\{\gamma_j\},\mathtt{C},\mathbf{d},e}{\arg\min} \ & \sum_{j=1}^{N}\gamma_j \\ \text{s.t. } & \gamma_j - g_j(\mathtt{C},\mathbf{d},e) > 0 \quad \forall\ j = 1,\ldots,N \\ & \gamma_j + g_j(\mathtt{C},\mathbf{d},e) > 0 \quad \forall\ j = 1,\ldots,N \\ & \gamma_j > 0 \quad \forall\ j = 1,\ldots,N \\ & \mathrm{trace}(\mathtt{C}) = 1 \end{aligned} \tag{6}$$

where g_j is a function of $\mathtt{C}$, $\mathbf{d}$, and e as defined in (5). We have used Sturm's SeDuMi [18], an open-source semidefinite programming solver, to obtain the global minimum. After the optimal ellipsoid variables ($\mathtt{C}$, $\mathbf{d}$, e) are obtained, these can be used to retrieve the sensor model variables ($\mathtt{K}$, $\mathbf{b}$) and the field directions $\{\hat{\mathbf{x}}_j\}$, as shown in Algorithm 1. (chol($\mathtt{C}$) denotes the Cholesky decomposition [20] of $\mathtt{C}$.)

Despite the yielded solution being a global minimum, it can incur bias as the minimized objective is not a physically meaningful quantity [21]. Hence, an additional step is required to better refine the solution, for instance, by minimizing the sum of sensor estimation errors $\|\mathbf{y}_j - \mathbf{m}_j\|$.

D. Robust Refinement of Sensor Variables and Field Directions

After an initial sensor model and 3-D field directions are obtained from Section II-C, they are further jointly refined to minimize a geometrically meaningful sensor estimation error. One thing to note is that each field direction must be normalized at all times during optimization. Such can be simultaneously handled by employing a robust nonlinear least-squares algorithm and incorporating a manifold optimization framework on top.



(a) (b)

Fig. 1. Ellipsoid fitting results on 454 data points obtained when calibrating near a stapler (MS1 from Table III); 442 blue dots denote inlier data points and 12 red crosses are anomalies. (a) Standard least-squares (L_2) solution yielding a visually incorrect model even from <3% outlier proportion. On the other hand, the L_1-norm solution in (b) is not so affected by the presence of outliers.

Our complete algorithm chart for sensor refinement can be found in Algorithm 2.

1) Problem Formulation: At time j with the field direction $\hat{\mathbf{x}}_j \in S^2$, the sensor estimation error $\mathbf{r}_j$ between the model estimate ($\mathtt{K}\hat{\mathbf{x}}_j + \mathbf{b}$) and the measurement ($\mathbf{m}_j$) is defined as

$$\mathbf{r}_j(\mathbf{k},\mathbf{b},\hat{\mathbf{x}}_j) := \mathtt{K}\hat{\mathbf{x}}_j + \mathbf{b} - \mathbf{m}_j \tag{7}$$

where $\mathbf{k} \in \mathbb{R}^6$ denotes a minimal representation of the upper triangular matrix $\mathtt{K}$, as shown in Table II.

As shown in Fig. 2, minimizing a simple L_2-norm objective $\sum_{j=1}^{N}\|\mathbf{r}_j(\mathbf{k},\mathbf{b},\hat{\mathbf{x}}_j)\|_2^2$ opens vulnerability to outlier data that can arise from versatile calibration conditions (see [22] for discussions). We therefore encapsulate $\mathbf{r}_j(\mathbf{k},\mathbf{b},\hat{\mathbf{x}}_j)$ with a robust kernel $\rho : \mathbb{R} \to \mathbb{R}$ such that we solve

$$\begin{aligned} \min_{\mathbf{k},\mathbf{b},\{\hat{\mathbf{x}}_j\}} \ & \sum_{j\in\Omega_t}\rho\left(\|\mathbf{r}_j(\mathbf{k},\mathbf{b},\hat{\mathbf{x}}_j)\|_2^2\right) \\ \text{s.t. } & \|\hat{\mathbf{x}}_j\|_2 = 1 \quad \forall\ j = 1,\ldots,N. \end{aligned} \tag{8}$$

This effectively discourages outlier data from changing the sensor model significantly.

2) Robust Kernels: An ideal kernel $\rho(\cdot)$ would effectively disregard outliers while preserving the behavior of the L_2 norm for the inliers. In practice, several choices exist [23], and we have tested four well-known selections, namely L_1, Huber, Cauchy and Geman–McClure (GM) kernels defined as follows:

$$\rho_{L_1}(s) = \sqrt{s} \tag{9}$$

$$\rho_{\text{Huber}}(s) = \begin{cases} s, & \text{if } s \le \tau^2 \\ 2\sqrt{s} - 1, & \text{if } s > \tau^2 \end{cases} \tag{10}$$

$$\rho_{\text{Cauchy}}(s) = \tau^2 \log\left(1 + \frac{s}{\tau^2}\right) \tag{11}$$

$$\rho_{\text{GM}}(s) = \frac{\tau^2}{\tau^2 + s} \tag{12}$$

where s is the L_2-norm cost and τ is the width (or radius) of the inlier-classified region. Note that all these kernels are second-order differentiable and have varying degrees of robustness, as shown in Fig. 2.

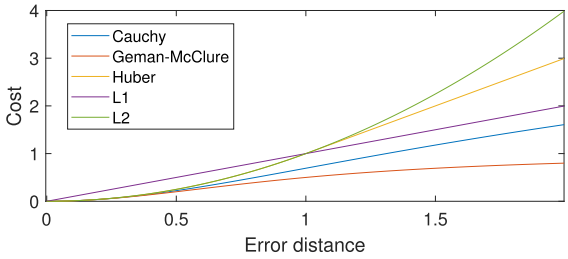


Fig. 2. Visualization of the robust kernels listed in Section II-D2. The standard L_2-norm kernel yields higher cost for residuals with larger error, thus being vulnerable to outliers not appropriately fitting the model. On the other hand, other kernels put less emphasis on larger residuals, gaining more robustness to outliers in exchange for creating more local minima due to lack of gradients (hence requiring a better initial solution).

3) Robustified Second-Order Subproblem: If we define a variable set $\boldsymbol{\theta} := [\mathbf{k}; \mathbf{b}; \hat{\mathbf{x}}_1; \ldots; \hat{\mathbf{x}}_n]$ and the corresponding update step as $\boldsymbol{\Delta\theta}$, the second-order surrogate cost function surface $q(\boldsymbol{\Delta\theta})$ used by a second-order optimizer can be defined as

$$\begin{aligned}\sum_{j\in\Omega_t} \rho\big(\|\mathbf{r}_j(\boldsymbol{\theta}+\boldsymbol{\Delta\theta})\|_2^2\big) &\approx \sum_{j=1}^{N} \rho\big(\|\mathbf{r}_j(\boldsymbol{\theta}) + \mathtt{J}_j\boldsymbol{\Delta\theta}\|_2^2\big) \\ &= \text{const} + \mathbf{g}^\top\boldsymbol{\Delta\theta} + \frac{1}{2}\boldsymbol{\Delta\theta}^\top\mathtt{H}\boldsymbol{\Delta\theta} =: q(\boldsymbol{\Delta\theta}) \end{aligned} \tag{13}$$

where $\mathtt{J}_j := \partial\mathbf{r}_j(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$ is the Jacobian of residual $\mathbf{r}_j(\boldsymbol{\theta})$ at $\boldsymbol{\theta}$ and $\mathbf{g}$ and $\mathtt{H}$ are the gradient and Hessian of the minimized objective (8) at $\boldsymbol{\theta}$.

For the Gauss–Newton algorithm, one simply needs to minimize $q(\boldsymbol{\Delta\theta})$ with respect to $\boldsymbol{\Delta\theta}$ in each iteration. Differentiating (13) and setting the derivative $q'(\boldsymbol{\Delta\theta}) = 0$ for optimality yields $\boldsymbol{\Delta\theta} = -\mathtt{H}^{-1}\mathbf{g}$. For the Levenberg–Marquardt (LM) algorithm [24], [25], there is an additional penalty term implicitly controlling the size of the update step. We solve

$$\arg\min_{\boldsymbol{\Delta\theta}} \; q(\boldsymbol{\Delta\theta}) + \lambda\|\boldsymbol{\Delta\theta}\|_2^2 \tag{14}$$

where the last term penalizes the size of the update step, implicitly setting a trust region within which the objective surface can be "trusted" as quadratic. The augmented update equation for $\boldsymbol{\Delta\theta}$ is

$$\boldsymbol{\Delta\theta} = -(\mathtt{H} + \lambda\mathtt{I})^{-1}\mathbf{g}. \tag{15}$$

The damping factor λ is adjusted in each iteration to reflect the size of this trust region—if the objective surface is mostly quadratic, then λ is decreased, behaving more like the standard Newton-based method. On the other hand, if the objective is not so quadratic, then λ is increased, discouraging large (potentially inaccurate) updates and more following down the route of gradient descent. $\mathbf{g}$ and $\mathtt{H}$ are derived in Section II-D4.

4) Deriving Gradient and the Gauss–Newton Matrix: For brevity of notations, we additionally define $\mathbf{r}_j := \mathbf{r}_j(\boldsymbol{\theta})$, $\rho' := \rho'(\|\mathbf{r}_j\|_2^2)$ (i.e., the first-order derivative of ρ), and $\rho'' := \rho''(\|\mathbf{r}_j\|_2^2)$ (the second-order derivative).

The gradient of (8) denoted as $\mathbf{g}$ can be written as

$$\begin{aligned}\mathbf{g} &:= \frac{\partial}{\partial\boldsymbol{\theta}}\sum_{j=1}^{N}\rho\big(\|\mathbf{r}_j(\boldsymbol{\theta})\|_2^2\big) \\ &= 2\sum_{j=1}^{N}\rho'\left[\frac{\partial\mathbf{r}_j}{\partial\boldsymbol{\theta}}\right]^\top\mathbf{r}_j =: 2\sum_{j=1}^{N}\rho'\mathtt{J}_j^\top\mathbf{r}_j \end{aligned} \tag{16}$$

where $\mathtt{J}_j \in \mathbb{R}^{3\times(9+3n)}$ is the Jacobian of the residual $\mathbf{r}_j$

$$\mathtt{J}_j := \begin{bmatrix}\dfrac{\partial\mathbf{r}_j}{\partial\mathbf{k}} & \dfrac{\partial\mathbf{r}_j}{\partial\mathbf{b}} & \dfrac{\partial\mathbf{r}_j}{\partial\hat{\mathbf{x}}_1} & \cdots & \dfrac{\partial\mathbf{r}_j}{\partial\hat{\mathbf{x}}_n}\end{bmatrix}. \tag{17}$$

By noting from [26] that $\text{vec}(\mathtt{AXB}) = (\mathtt{B}^\top\otimes\mathtt{A})\,\text{vec}\,\mathtt{X}$, we can write $\mathtt{K}\hat{\mathbf{x}}_j = \text{vec}(\mathtt{K}\hat{\mathbf{x}}_j) = (\hat{\mathbf{x}}_j^\top\otimes\mathtt{I})\,\text{vec}(\mathtt{K}) = (\hat{\mathbf{x}}_j^\top\otimes\mathtt{I})\Pi\mathbf{k}$, yielding

$$\frac{\partial\mathbf{r}_j}{\partial\mathbf{k}} = \big(\hat{\mathbf{x}}_j^\top\otimes\mathtt{I}\big)\Pi \tag{18}$$

$$\frac{\partial\mathbf{r}_j}{\partial\mathbf{b}} = \mathtt{I} \tag{19}$$

$$\frac{\partial\mathbf{r}_j}{\partial\hat{\mathbf{x}}_l} = \begin{cases}\mathtt{K}, & \text{if } j = l \\ \mathtt{0}, & \text{otherwise.}\end{cases} \tag{20}$$

The Hessian $\mathtt{H} := \partial^2 f/\partial\boldsymbol{\theta}^2$ is then

$$\begin{aligned}\mathtt{H} &:= 2\sum_{j=1}^{N} 2\rho''\big(\mathtt{J}_j^\top\mathbf{r}_j\mathbf{r}_j^\top\mathtt{J}_j\big) + \rho'\mathtt{J}_j^\top\mathtt{J}_j + \rho'\left[\frac{\partial\mathtt{J}_j}{\partial\boldsymbol{\theta}}\right]^\top\mathbf{r}_j \\ &\approx 2\sum_{j=1}^{N}\mathtt{J}_j^\top\big(\rho' + 2\rho''\mathbf{r}_j\mathbf{r}_j^\top\big)\mathtt{J}_j \end{aligned} \tag{21}$$

where the approximation discards the second-order derivative and is also known as the Gauss–Newton matrix [22].

5) Trigg's Correction for the Robust Norm Implementation: While it is possible to compute (21) naively, Triggs *et al.* [27] showed appropriate reweighting of the nonrobust residual vectors $\{\mathbf{r}_j\}$ and the corresponding Jacobians $\{\mathtt{J}_j\}$ allows employment of any second-order optimizer to solve the robust problem (8). More specifically, as illustrated by Agarwal *et al.* [22], one can compute reweighted residuals $\{\tilde{\mathbf{r}}_j\}$ and Jacobians $\{\tilde{\mathtt{J}}_j\}$ as

$$\tilde{\mathbf{r}}_j := \frac{\sqrt{\rho'}}{1-\alpha_j}\mathbf{r}_j, \tag{22}$$

$$\tilde{\mathtt{J}}_j := \sqrt{\rho'}\left(1 - \alpha_j\frac{\mathbf{r}_j\mathbf{r}_j^\top}{\|\mathbf{r}_j\|_2^2}\right)\mathtt{J}_j \tag{23}$$

where α_j is a solution of

$$\frac{1}{2}\alpha_j^2 - \alpha_j - \frac{\rho''}{\rho'}\|\mathbf{r}_j\|_2^2 = 0 \tag{24}$$

for all $j = 1, \ldots, N$. It is then straightforward to see

$$\mathbf{g} = 2\sum_{j=1}^{N}\rho'\mathtt{J}_j^\top\mathbf{r}_j = 2\sum_{j=1}^{N}\tilde{\mathtt{J}}_j^\top\tilde{\mathbf{r}}_j \tag{25}$$

$$\mathtt{H} \approx 2\sum_{j=1}^{N}\mathtt{J}_j^\top\big(\rho' + 2\rho''\mathbf{r}_j\mathbf{r}_j^\top\big)\mathtt{J}_j = 2\sum_{j=1}^{N}\tilde{\mathtt{J}}_j^\top\tilde{\mathtt{J}}_j. \tag{26}$$

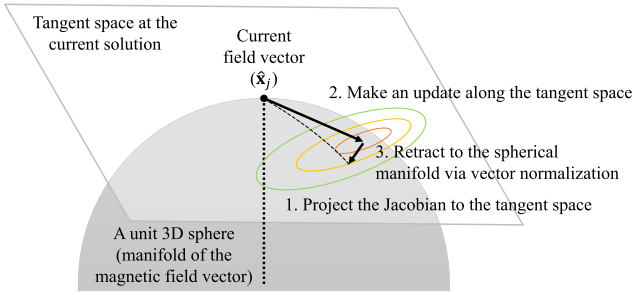


Fig. 3. Illustration of the Riemannian manifold optimization technique for a field direction vector $\hat{\mathbf{x}}_j \in S^2$. First, the Jacobian with respect to $\hat{\mathbf{x}}_j$ is projected to the tangent space of $\hat{\mathbf{x}}_j$, which can be regarded as drawing a flattened map of gradients. Second, an update is made along the tangent space. Finally, the updated field is retracted back to the unit sphere through vector normalization.

6) Manifold Optimization: We are still left with the problem of enforcing $\|\hat{\mathbf{x}}_j\|_2 = 1$ for each field direction vector $\hat{\mathbf{x}}_j$. While some prior works have considered relaxing the constraint by adding $\nu(\|\hat{\mathbf{x}}_j\|_2 - 1)^2$ or $\nu(\|\hat{\mathbf{x}}_j\|_2^2 - 1)^2$ as a penalty term [8] for some $\nu \in \mathbb{R}^+$, this does not strictly guarantee each field vector to be unit normed unless $\nu \to \infty$. Instead, we apply a Riemannian manifold optimization framework [16] (also known as local parameterization [22]) to consider the 3-D spherical manifold of the field vectors. As the mathematical derivations are well illustrated in Absil *et al.*'s book [16], we just summarize the implementation details next.

First, the Jacobian with respect to each field $\hat{\mathbf{x}}_j$ is projected to the tangent space at $\hat{\mathbf{x}}_j$. If we define such portion of Trigg's Jacobian $\tilde{\mathtt{J}}$ as $\tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j}$ (from the 10th column of $\tilde{\mathtt{J}}$ to the end), the projected Jacobian is

$$\tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j} \leftarrow \tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j}\left(\mathtt{I} - \hat{\mathbf{x}}_j\hat{\mathbf{x}}_j^\top\right). \tag{27}$$

This ensures zero gradient along the field vector $\hat{\mathbf{x}}_j$, preventing degenerate updates along the current field direction.

Second, a penalty term proposed by Okatani *et al.* [28] is added to the LM subproblem (14) to ensure that updates are made along the tangent space. More specifically, the modified subproblem becomes

$$\underset{\boldsymbol{\Delta\theta}}{\arg\min}\ q(\boldsymbol{\Delta\theta}) + \lambda\|\boldsymbol{\Delta\theta}\|_2^2 + \beta\sum_{j=1}^{N}\|\hat{\mathbf{x}}_j\Delta\hat{\mathbf{x}}_j\|_2^2 \tag{28}$$

which can be solved through least-squares formulation per iteration. Solving (28) yields the augmented solution

$$\boldsymbol{\Delta\theta} = -(\mathtt{H} + \lambda\mathtt{I} + \beta\,\mathtt{D})^{-1}\mathbf{g} =: -\tilde{\mathtt{H}}^{-1}\mathbf{g} \tag{29}$$

where $\mathtt{D}$ is a block-diagonal matrix defined as

$$\begin{bmatrix} 0 & & \\ & \mathtt{D}_1 & \\ & & \ddots \\ & & & \mathtt{D}_N \end{bmatrix} = \begin{bmatrix} 0 & & \\ & \hat{\mathbf{x}}_1\hat{\mathbf{x}}_1^\top & \\ & & \ddots \\ & & & \hat{\mathbf{x}}_N\hat{\mathbf{x}}_N^\top \end{bmatrix}. \tag{30}$$

The first block is 0 as no manifold constraint is applied to $\mathbf{p} := [\mathbf{k}; \mathbf{b}]$. In our implementation, β is set to 1.

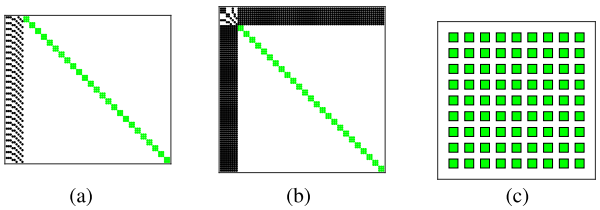


Fig. 4. Structures of the derivative matrices used during refinement of sensor variables. The original Jacobian and Hessian matrices are sparse, as each residual $\mathbf{r}_j$ only depends on the corresponding field vector $\hat{\mathbf{x}}_j$ and not the others. In Fig. 4(a), the left part in black corresponds to $\mathtt{J}_{\mathbf{p}} := [\mathtt{J}_{\mathbf{k}}, \mathtt{J}_{\mathbf{b}}]$, and the right block diagonal part in green corresponds to $\mathtt{J}_{\hat{\mathbf{x}}} := [\mathtt{J}_{\hat{\mathbf{x}}_1}, \cdots, \mathtt{J}_{\hat{\mathbf{x}}_N}]$. The approximate Hessian (Gauss–Newton JTJ) matrix yielded from $\mathtt{J}$ has a special partly block diagonal structure, as shown in Fig. 4(b). Naively inverting Fig. 4(b) requires $O(N^3)$ computations, where N is the number of data points. By using the Schur complement trick in Section II-D7, a reduced system of 9×9 is formed as shown in Fig. 4(c) [see the left-hand side of (33)] with only $O(N)$ computations, improving the per-iteration efficiency of the refinement step. (a) Original Jacobian. (b) Original JTJ. (c) Reduced JTJ.

Finally, the updated field direction vector $\hat{\mathbf{x}}_j + \Delta\hat{\mathbf{x}}_j$, which no longer lies on the unit 3-D sphere, is retracted back to its original manifold through normalization

$$\hat{\mathbf{x}}_j \leftarrow \frac{\hat{\mathbf{x}}_j + \Delta\hat{\mathbf{x}}_j}{\|\hat{\mathbf{x}}_j + \Delta\hat{\mathbf{x}}_j\|_2}. \tag{31}$$

These procedures are visually shown in Fig. 3.

7) Schur Complement Trick for Efficient Implementation: In this section, we illustrate the well-known Schur complement trick [27] for efficient implemention of model refinement. As each residual $\mathbf{r}_j$ only depends on that particular time's field direction $\hat{\mathbf{x}}_j$, $\mathtt{J}_j$ contains many zeros as shown in Fig. 4(a), leading to a largely sparse Gauss–Newton matrix, as shown in Fig. 4(b). This allows use of the Schur complement trick [27] for efficient computation of (29).

If we define $\mathbf{p} := [\mathbf{k}; \mathbf{b}]$, then (29) can be written as

$$\begin{bmatrix} \tilde{\mathtt{H}}_{\mathbf{p},\mathbf{p}} & \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_1} & \cdots & \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_N} \\ \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_1}^\top & \tilde{\mathtt{H}}_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_1} & & \\ \vdots & & \ddots & \\ \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_N}^\top & & & \tilde{\mathtt{H}}_{\hat{\mathbf{x}}_N,\hat{\mathbf{x}}_N} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta\mathbf{p}} \\ \boldsymbol{\Delta\hat{\mathbf{x}}}_1 \\ \vdots \\ \boldsymbol{\Delta\hat{\mathbf{x}}}_N \end{bmatrix} = - \begin{bmatrix} \mathbf{g}_{\mathbf{p}} \\ \mathbf{g}_{\hat{\mathbf{x}}_1} \\ \vdots \\ \mathbf{g}_{\hat{\mathbf{x}}_N} \end{bmatrix}.$$

Note that the bottom-right segment of $\tilde{\mathtt{H}}$ is block-diagonal and the off-diagonals are not tilded as they are not augmented by the diagonal terms $\lambda\mathtt{I}$ and $\mathtt{D}$ in (29). By further defining $\hat{\mathbf{x}} := [\hat{\mathbf{x}}_1; \cdots; \hat{\mathbf{x}}_N]$, above can be written as

$$\begin{bmatrix} \tilde{\mathtt{H}}_{\mathbf{p},\mathbf{p}} & \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}} \\ \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}}^\top & \tilde{\mathtt{H}}_{\hat{\mathbf{x}},\hat{\mathbf{x}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta\mathbf{p}} \\ \boldsymbol{\Delta\hat{\mathbf{x}}} \end{bmatrix} = - \begin{bmatrix} \mathbf{g}_{\mathbf{p}} \\ \mathbf{g}_{\hat{\mathbf{x}}} \end{bmatrix}, \tag{32}$$

where $\tilde{\mathtt{H}}_{\hat{\mathbf{x}},\hat{\mathbf{x}}} \in \mathbb{R}^{3N\times 3N}$ is the aforementioned block diagonal matrix with each block being 3×3. Solving this naively requires $O((9 + 3N)^3) = O(N^3)$ computation due to the inversion of $\tilde{\mathtt{H}}$. However, if we write (32) as a system of linear equations, (32) yields

$$\left(\tilde{\mathtt{H}}_{\mathbf{p},\mathbf{p}} - \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}}\tilde{\mathtt{H}}_{\hat{\mathbf{x}},\hat{\mathbf{x}}}^{-1}\mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}}^\top\right)\boldsymbol{\Delta\mathbf{p}} = -\mathbf{g}_{\mathbf{p}} + \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}}\tilde{\mathtt{H}}_{\hat{\mathbf{x}},\hat{\mathbf{x}}}^{-1}\mathbf{g}_{\hat{\mathbf{x}}} \tag{33}$$

$$\tilde{\mathtt{H}}_{\hat{\mathbf{x}},\hat{\mathbf{x}}}\boldsymbol{\Delta\hat{\mathbf{x}}} = -\mathbf{g}_{\hat{\mathbf{x}}} - \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}}^\top\boldsymbol{\Delta\mathbf{p}} \tag{34}$$

which can be solved sequentially. Since the dimension of $\tilde{\mathtt{H}}_{\mathbf{p},\mathbf{p}} \in \mathbb{R}^{9\times 9}$ is smaller than $\tilde{\mathtt{H}}_{\hat{\mathbf{x}},\hat{\mathbf{x}}}$ (assuming $N > 3$), the computation bottleneck is down to inverting $\tilde{\mathtt{H}}_{\hat{\mathbf{x}},\hat{\mathbf{x}}}$. Since the inverse

Algorithm 2 Robust Nonlinear Optimization of Sensor Intrinsics ($\mathtt{K}, \mathbf{b}$) and 3-D Field Directions $\{\hat{\mathbf{x}}_j\}$

Inputs: initial sensor model ($\mathtt{K}, \mathbf{b}$), field directions $\{\hat{\mathbf{x}}_j\}$ and measurements $\{\mathbf{m}_j\}$
Initialize damping factor $\lambda \leftarrow 10$.
$f_{\text{best}} \leftarrow \sum_{j=1}^{N} \rho(\|\mathtt{K}\hat{\mathbf{x}}_j + \mathbf{b} - \mathbf{m}_j\|_2^2)$
repeat
  Initialize derivatives $\mathbf{g_p} \leftarrow 0$ and $\mathtt{H}_{\mathbf{p},\mathbf{p}} \leftarrow 0$.
  for $j = 1, \ldots, N$ **do**
    $\mathbf{r}_j \leftarrow \mathtt{K}\hat{\mathbf{x}}_j + \mathbf{b} - \mathbf{m}_j$
    Compute $\mathtt{J}_j$ from Sec. II-D4
    Obtain α_j by solving (24)
    Compute $\tilde{\mathbf{r}}_j$ from (22)
    Compute $\tilde{\mathtt{J}}_j = [\tilde{\mathtt{J}}_{j,\mathbf{p}}, \tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_1}, \cdots, \tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_N}]$ from (23)
    Retrieve $\tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j}$ from $\tilde{\mathtt{J}}_j$.
    $\tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j} \leftarrow \tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j}(\mathtt{I} - \hat{\mathbf{x}}_j\hat{\mathbf{x}}_j^\top)$
    $f_{\text{best}} \leftarrow f_{\text{best}} + \rho(\|\mathbf{r}_j\|_2^2)$
    $\mathbf{g}_{\hat{\mathbf{x}}_j} \leftarrow 2\tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j}^\top \tilde{\mathbf{r}}_j$
    $\mathbf{g_p} \leftarrow \mathbf{g_p} + 2\tilde{\mathtt{J}}_{j,\mathbf{p}}^\top \tilde{\mathbf{r}}_j$
    $\mathtt{H}_{\hat{\mathbf{x}}_j,\hat{\mathbf{x}}_j} \leftarrow 2\tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j}^\top \tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j}$
    $\mathtt{D}_j \leftarrow \hat{\mathbf{x}}_j\hat{\mathbf{x}}_j^\top$ (manifold constraint)
    $\mathtt{H}_{\hat{\mathbf{x}}_j,\hat{\mathbf{x}}_j} \leftarrow \mathtt{H}_{\hat{\mathbf{x}}_j,\hat{\mathbf{x}}_j} + \mathtt{D}_j$
    $\mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_j} \leftarrow 2\tilde{\mathtt{J}}_{j,\mathbf{p}}^\top \tilde{\mathtt{J}}_{j,\hat{\mathbf{x}}_j}$
    $\mathtt{H}_{\mathbf{p},\mathbf{p}} \leftarrow \mathtt{H}_{\mathbf{p},\mathbf{p}} + 2\tilde{\mathtt{J}}_{j,\mathbf{p}}^\top \tilde{\mathtt{J}}_{j,\mathbf{p}}$
  end for
  $\mathbf{g}_{\hat{\mathbf{x}}} \leftarrow [\mathbf{g}_{\hat{\mathbf{x}}_1}^\top, \cdots, \mathbf{g}_{\hat{\mathbf{x}}_N}^\top]^\top$
  $\mathtt{H}_{\hat{\mathbf{x}},\hat{\mathbf{x}}} \leftarrow \mathtt{blkdiag}(\mathtt{H}_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_1}, \cdots, \mathtt{H}_{\hat{\mathbf{x}}_N,\hat{\mathbf{x}}_N})$ (MATLAB)
  $\mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}} \leftarrow [\mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_1}, \cdots, \mathtt{H}_{\mathbf{p},\hat{\mathbf{x}}_N}]$
  repeat
    $\tilde{\mathtt{H}}_{\mathbf{p},\mathbf{p}} \leftarrow \mathtt{H}_{\mathbf{p},\mathbf{p}} + \lambda\mathtt{I}$ (trust region damping)
    $\tilde{\mathtt{H}}_{\hat{\mathbf{x}},\hat{\mathbf{x}}} \leftarrow \mathtt{H}_{\hat{\mathbf{x}},\hat{\mathbf{x}}} + \lambda\mathtt{I}$ (trust region damping)
    Compute $\boldsymbol{\Delta}\mathbf{p}$ from (33) and retrieve $\Delta\mathtt{K}$ and $\boldsymbol{\Delta}\mathbf{b}$.
    Compute $\boldsymbol{\Delta}\hat{\mathbf{x}}$ from (34) and retrieve $\{\boldsymbol{\Delta}\hat{\mathbf{x}}_j\}$.
    $f \leftarrow \sum_j \rho(\|\mathbf{r}_j(\mathtt{K} + \Delta\mathtt{K}, \mathbf{b} + \boldsymbol{\Delta}\mathbf{b}, \hat{\mathbf{x}}_j + \boldsymbol{\Delta}\hat{\mathbf{x}}_j)\|_2^2)$
    if $f < f_{\text{best}}$ (i.e. found a better model) **then**
      $[\mathtt{K}, \mathbf{b}] \leftarrow [\mathtt{K} + \Delta\mathtt{K}, \mathbf{b} + \boldsymbol{\Delta}\mathbf{b}]$ (update model)
      $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \Delta\hat{\mathbf{x}}$ (update fields)
      $\lambda \leftarrow \max(\lambda/10, 10^{-14})$ (larger trust region)
      `break`
    else
      $\lambda \leftarrow \min(10\lambda, 10^{14})$ (smaller trust region)
    end if
  until max. # trials reached
until max. # iterations or function tolerance reached
Outputs: sensor model ($\mathtt{K}, \mathtt{b}$) and field directions $\{\hat{\mathbf{x}}_j\}$

of a block diagonal matrix can be computed from the inverses of individual blocks, this is only an $O(N \times 3^2) = O(N)$ computation for our case.

For empirical verification, we compared the runtime of our Schur complement trick implementation against that of a naive robust LM algorithm. For 1000 measurements, the algorithm incorporating the Schur complement trick reported 46–60 ms per iteration, which is a 4.5–7 factor of speed up when compared with the naive version reporting 288–320 ms.

III. Experimental Results

We tested our algorithm by comparing its performance against baseline and state-of-the-art self-calibration algorithms on synthetic and real magnetometer measurements. For this purpose, we have tested variants of our method proposed in Section II by employing different robust kernels for refinement (L_1, GM, Huber, and Cauchy) and reimplemented all the algorithms listed in Table I.

All the experiments were carried out in MATLAB R2019b on a PC with an AMD R2700X CPU and 32-GB DDR4 RAM.

A. Algorithm Settings

For a fair comparison, we set each iterative algorithm's maximum number of iterations to 300 and the relative function tolerance value (at which the algorithm terminates) to 10^{-6}. Data normalization from Section II-B is applied to all methods. We modified all the algorithms to output an upper triangular sensor matrix (i.e., $\mathtt{A} = \mathtt{K}$), which can be done by RQ-decomposing $\mathtt{A}$ without affecting the original performance.

For algorithms requiring L_2-norm algebraic ellipsoid fitting, we implemented an SVD-based method described in the work of Kok and Schön [11]. For algorithms involving semidefinite programming, we used Sturm's SeDuMi solver [18] available in the CVX library [29]. In addition, for algorithms requiring nonlinear optimization, we employed the same LM [22], [24], [25] algorithm to minimize the effect of other implementational factors.

All our algorithms requiring the use of robust kernels with variable kernel widths (Huber, Cauchy, and GM) have set their widths (τ) to $\sqrt{3}$, assuming on average up to 1 unit error per sensor axis for an inlier data point.

B. Simulation Study

We have designed our synthetic experiment to be able to observe each algorithm's performance across a range of noise level and proportion of anomalies. To achieve this, we generated synthetic data with varying additive Gaussian noise standard deviation (σ) and proportion of outliers (η), ran each algorithm from Table I on each experimental condition (σ, η) until convergence, and reported each algorithm's deviation in model variables from ground truth. The entire process was repeated multiple times to gain consistency in results. More details are provided in the following.

1) Generating Synthetic Data per Experimental Condition: Simulation data generation depends on the input experimental conditions, namely, the Gaussian noise standard deviation σ and the proportion of anomalies η.

First, a synthetic magnetometer model ($\mathtt{K}, \mathbf{b}$) mimicing a real-world three-axis magnetometer is created using the equations

$$\mathtt{K} = \sigma_{\mathtt{K}} \left(\mathtt{I} + 0.1 \begin{bmatrix} \Delta k_1 & \Delta k_2 & \Delta k_4 \\ & \Delta k_3 & \Delta k_5 \\ & & \Delta k_6 \end{bmatrix} \right) \tag{35}$$

$$\mathbf{b} = \sigma_{\mathbf{b}} \left(\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} \Delta b_1 \\ \Delta b_2 \\ \Delta b_3 \end{bmatrix} \right) \tag{36}$$

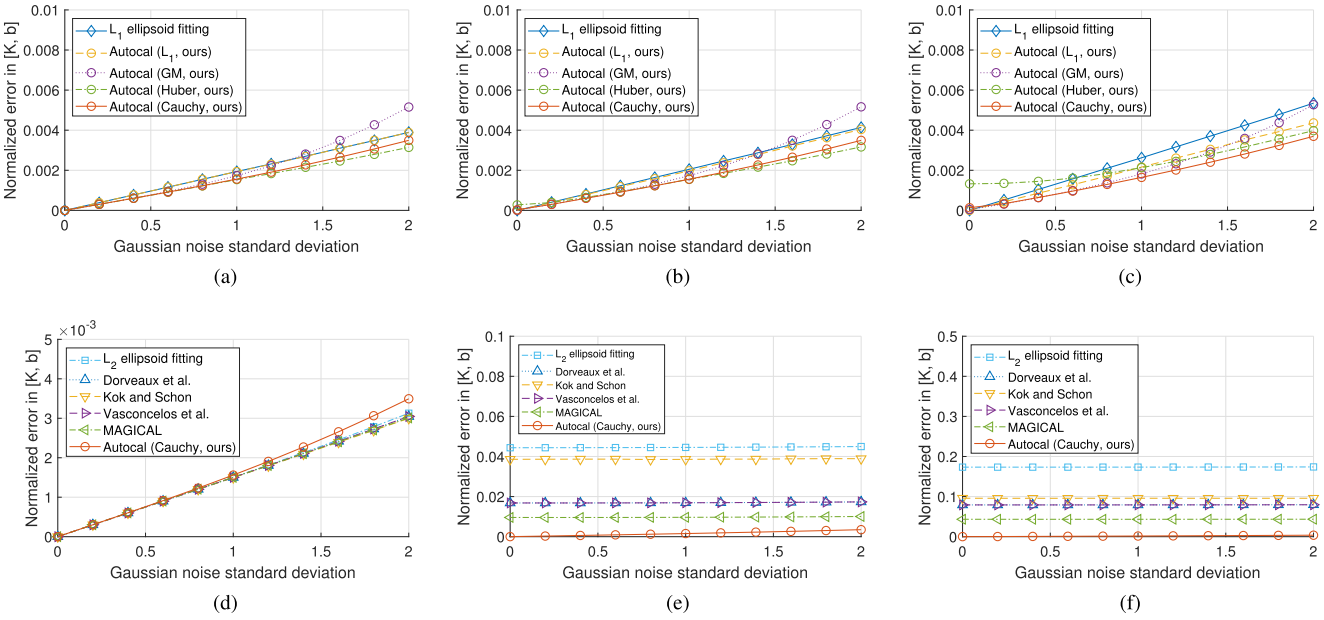


Fig. 5. Results on synthetic data as a function the standard deviation (σ) of additive white Gaussian noise. Average normalized sensor estimation error ($\|[\mathrm{K}, \mathbf{b}] - [\mathrm{K}_{gt}, \mathbf{b}_{gt}]\|_F / \|[\mathrm{K}_{gt}, \mathbf{b}_{gt}]\|_F$, lower the better) is reported for each setting of σ when a specific proportion of outliers (η) exists. The top row illustrates the performance of our method variants, whereas bottom compares our Cauchy version against baseline and state-of-the-art algorithms reviewed in Table I. The left column presents the results when no anomaly is included, the middle column shows when η is 1%, and the right column corresponds to $\eta = 10\%$. (a) No outliers (ours). (b) 1% outliers (ours). (c) 10% outliers (ours). (d) No outliers (versus others). (e) 1% outliers (versus others). (f) 10% outliers (versus others).

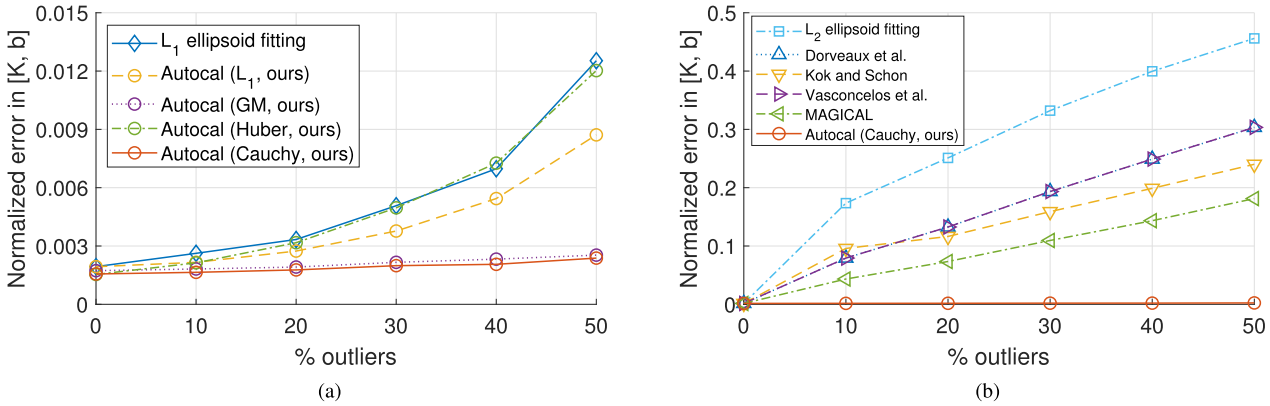


Fig. 6. Results on synthetic data as a function of number of outliers. Average normalized sensor estimation error ($\|[\mathrm{K}, \mathbf{b}] - [\mathrm{K}_{gt}, \mathbf{b}_{gt}]\|_F / \|[\mathrm{K}_{gt}, \mathbf{b}_{gt}]\|_F$, lower the better) is reported for each outlier proportion setting (η) on synthetic data. The standard deviation of additive Gaussian noise (σ) is fixed at 1.0, which is similar to that obtained from real magnetometer readings. (a) Between our algorithms. (b) Versus others.

where each Δk_i is drawn from $\mathcal{N}(0, 1)$. After inspecting the average scale of sensor readings from a real strapdown magnetometer, we have set σ_{K} to 100 and $\sigma_{\mathbf{b}}$ to 20.

Second, the associated sensor readings are generated using the above sensor model $(\mathrm{K}, \mathbf{b})$ and a set of 1000 arbitrarily sampled field directions on a unit 3-D sphere $\{\hat{\mathbf{x}}_j\}$, depicting a magnetometer being randomly rotated for autocalibration. To simulate real-world environment, sensor outputs $\{\mathbf{y}_j\}$ are perturbed by additive white Gaussian noise of standard deviation σ, and some heavy anomalous errors are added to a fraction η of measurements to generate outliers. More specifically, if the jth measurement $\mathbf{m}_j$ is set as an inlier data point, it is generated by the equation

$$\mathbf{m}_j = \mathrm{K}\hat{\mathbf{x}}_j + \mathbf{b} + \boldsymbol{\varepsilon}_j \tag{37}$$

where $(\mathrm{K}, \mathbf{b})$ are the synthetic sensor variables, $\hat{\mathbf{x}}_j \in S^2$ is the field direction at time j, and $\boldsymbol{\varepsilon}_j \sim \mathcal{N}(\mathbf{0}, \sigma\,\mathrm{I})$ is the white Gaussian noise. On the other hand, if $\mathbf{m}_j$ is set as an outlier, then we additionally add a heavy perturbation term $\boldsymbol{\xi}$ such that

$$\mathbf{m}_j \leftarrow \mathbf{m}_j + \boldsymbol{\xi}_j \tag{38}$$

where each element of $\boldsymbol{\xi}_j \in \mathbb{R}^3$ is sampled from a uniform distribution $\mathcal{U}(-\zeta, \zeta)$ with ζ being the maximum scale of the outlier noise. After probing through the scale of outliers from real data, ζ is set to 100.

2) Experimental Procedure: For the synthetic experiments, we varied the Gaussian noise standard deviation (σ) between 0 and 2 in the sensor measurement unit and the proportion of outliers (η) between 0% and 50%. Since the standard deviation of noise estimated from outlier-free real data (M1 and M2 in Table III) is 0.69 and the proportion of outliers across outlier-inclusive data sets (MP1, MP2, MS1, MS2, and MS3 in Table III) ranges between 2.6% and 6.9%, we believe that these ranges are wide enough to cover real-world settings.

We ran each of the algorithms in Table I on each experimental condition (σ, η) until convergence and reported deviation of the estimated model variables $(\mathtt{K}, \mathbf{b})$ from ground truth $(\mathtt{K}_{\text{gt}}, \mathbf{b}_{\text{gt}})$. More specifically, we computed the normalized model error defined as

$$\frac{\|[\mathtt{K} \quad \mathbf{b}] - [\mathtt{K}_{\text{gt}} \quad \mathbf{b}_{\text{gt}}]\|_F}{\|[\mathtt{K}_{\text{gt}} \quad \mathbf{b}_{\text{gt}}]\|_F}. \tag{39}$$

We repeated the above procedure 25 times with different synthetic data to obtain an average of normalized model deviations. To keep things fair and consistent, we made sure that all algorithms run on an identical set of synthetic measurements at each experimental condition (σ, η).

3) Results on Synthetic Data: We have produced two representative figures after carrying out the above experiments.

Fig. 5 shows each algorithm's calibration accuracy as a function of additive Gaussian noise standard deviation (σ) given a specific proportion of outliers (η). The top row consists of intraclass comparisons between our algorithms each employing a different robust kernel, and the bottom row presents interclass comparisons against other baseline and state-of-the-art methods. The left column is with zero outliers (heavy anomalies), the middle column is with 1% outliers, and the right column is with 10% outliers all generated, as shown in Section III-B1.

Fig. 6 shows each algorithm's performance as a function of outlier proportions (η) with $\sigma = 1$. Intraclass (between our algorithm variants) and interclass comparisons are shown on the left- and the right-hand sides, respectively.

1) *Intraclass Comparisons:* It can be seen from the top row of Fig. 5 that the performance variations between different robust kernels are not significant for up to $\eta = 10\%$, although Huber performs better for lower proportion of outliers (η) and Cauchy performs better for higher η. Fig. 6 shows much widening gap between the accuracies achieved by Huber and Cauchy as η increases. We believe that this is because Cauchy is more robust than Huber, and it puts less emphasis on the outlier-classified data points as shown in Fig. 2, leading to improved accuracy as η increases.
   On the other hand, the GM kernel, which is even more robust than Cauchy, shows degrading accuracy in Fig. 5 as σ increases. We conjecture that this is due to the Gaussian noise level (σ) surpassing the size of the kernel width (set to $\sqrt{3}$, i.e., optimal for $\sigma = 1$), meaning that data points that are not actually anomalies are frequently classified as outliers in this case, leading to a biased model estimate that is less accurate. When σ is kept at 1.0, GM shows comparable performance to Cauchy for a wide tested range of η in Fig. 6.
   From Figs. 5 and 6, we determined that Cauchy performs most stably overall across a range of outlier proportions and Gaussian noise level and decided to select it as our representative algorithm for comparison against other methods.
2) *Interclass Comparisons:* The bottom row of Figs. 5 and 6(b) illustrates the performance comparison between ours and other methods listed in Table I.

TABLE III

LIST OF REAL SEQUENCES OBTAINED USING AN IMU. THE NUMBER OF OUTLIERS HAS BEEN APPROXIMATELY COMPUTED BY CALIBRATING THE SENSOR USING OUR METHOD FROM SECTION II AND COUNTING THE NUMBER OF OBSERVATIONS WITH ERRORS LARGER THAN FIVE TIMES THE WIDTH OF THE EMPLOYED ROBUST KERNELS ($\sqrt{3}$)

Sensor type	Sequence ID	# measurements (N)	# outliers (fraction η)	
magnetometer	M1	326	0	(0.0 %)
	M2	426	0	(0.0 %)
	MP1	454	15	(3.3 %)
	MP2	422	29	(6.9 %)
	MS1	454	12	(2.6 %)
	MS2	476	28	(5.9 %)
	MS3	421	18	(4.3 %)
accelero -meter	A1	362	0	(0.0 %)
	A2	589	7	(1.2 %)
	AO1	492	65	(13.2 %)
	AO2	477	60	(12.6 %)

When no outlier is present (i.e., $\eta = 0$), Fig. 5(d) shows similar results for all algorithms up to the Gaussian noise standard deviation (σ) of 1.2. Then, our Autocal performs slightly worse for higher σ than other methods. Again, similar to the reason for GM underperforming for higher σ with extremely low proportion of outliers (η), we believe that this is caused by the Gaussian noise level (σ) surpassing the width of the Cauchy kernel ($\sqrt{3}$, optimal for $\sigma = 1$), making the algorithm to downweight inlier data points with large noise, effectively yielding a suboptimal solution.
However, when η increases to just 1%, our Autocal outperforms other algorithms across tested range of σ, as shown in Fig. 5(e). This difference becomes more distinct as the outlier proportion increases to 10% in Fig. 5(f). Fig. 6(b) verifies this phenomenon. Note that the choice of robust kernel does not change the ranking of our algorithm, and this clearly demonstrates that robust optimization helps to retrieve a more accurate sensor model even in the presence of just a few outliers. Aside from our Autocal, it is worth noting that Papafotis and Sotiriadis' MAGICAL performs better than other L_2-norm-based methods. This indicates that minimizing the sensor estimation error yields a more accurate model than when minimizing either algebraic or geometric ellipsoid fitting errors.

3) *Ablation Study:* In Figs. 5 and 6, we have also included the result of sole L_1-norm ellipsoid fitting without refinement for ablation study. When comparing between L_1-norm ellipsoid fitting and our L_1-norm two-step algorithm, our method shows improvement over the ellipsoid fitting results, especially as the proportion of outliers (η) increases. This again demonstrates that minimizing a geometrically meaningful error distance is important for improving solution quality.

C. Real Experiments

We have also compared the performance of our algorithms, baselines, and state of the arts on a set of real IMU data.

TABLE IV

DEVIATION OF SENSOR VARIABLES ($\mathtt{K}$, $\mathbf{b}$) FROM THE REFERENCE MODEL ($\mathtt{K}_{\text{ref}}$, $\mathbf{b}_{\text{ref}}$) OBTAINED BY SELF-CALIBRATING ON THE M1 SEQUENCE (A1 FOR ACCELEROMETERS). NORMALIZED SENSOR DEVIATION ERROR $\|[\mathtt{K}, \mathbf{b}] - [\mathtt{K}_{\text{ref}}, \mathbf{b}_{\text{ref}}]\|_F / \|[\mathtt{K}_{\text{ref}}, \mathbf{b}_{\text{ref}}]\|_F$ IS REPORTED

Algorithm (↓), Dataset (→)	MAGNETOMETER						ACCELEROMETER		
	M2	MP1	MP2	MS1	MS2	MS3	A2	AO1	AO2
L_2-norm ellipsoid fitting (least squares)	2.0%	10.7%	11.0%	48.8%	14.7%	5.3%	0.6%	15.1%	11.6%
Dorveaux et al. [8]	55.8%	97.4%	73.8%	36.8%	136.4%	34.3%	0.6%	8.2%	6.0%
Kok and Schön [11]	2.1%	7.9%	6.7%	26.6%	9.0%	3.2%	**0.5%**	8.7%	9.8%
Vasconcelos et al. [1]	2.0%	34643.9%	32523.4%	41604.4%	5.6%	3.0%	0.6%	8.2%	6.0%
MAGICAL [7]	2.1%	2.9%	135.5%	205.5%	4.0%	2.7%	0.5%	4.1%	4.1%
L_1-norm ellipsoid fitting (SDP)	2.5%	**0.8%**	1.9%	6.4%	1.3%	1.6%	0.6%	1.2%	1.1%
Autocal (L_1, **ours**)	1.9%	2.2%	3.8%	2.0%	1.4%	1.4%	0.6%	1.1%	1.0%
Autocal (Cauchy, **ours**)	2.0%	2.1%	2.2%	1.9%	1.3%	1.3%	0.6%	1.3%	**0.7%**
Autocal (Geman-McClure, **ours**)	**1.9%**	2.6%	**1.4%**	1.9%	**1.2%**	**1.2%**	1.0%	1.9%	1.2%
Autocal (Huber, **ours**)	2.0%	1.9%	5.3%	**1.8%**	1.6%	1.6%	0.5%	**0.9%**	0.9%

TABLE V

AVERAGE SENSOR ESTIMATION ERROR (DEFINED IN SECTION III-C2) ACHIEVED BY DIFFERENT CALIBRATION ALGORITHMS. ON EACH DATA SET, EACH ALGORITHM LEARNS A MODEL ($\mathtt{K}$, $\mathbf{b}$), WHICH IS THEN TESTED ON THE OUTLIER-FREE M1 SEQUENCE (A1 FOR ACCELEROMETERS) FROM TABLE III

Algorithm (↓), Dataset (→)	MAGNETOMETER						ACCELEROMETER		
	M2	MP1	MP2	MS1	MS2	MS3	A2	AO1	AO2
L_2-norm ellipsoid fitting (least squares)	0.73	2.00	1.89	11.01	2.75	1.36	1.35	9.00	6.14
Dorveaux et al. [8]	18.08	13.19	13.07	20.95	25.51	17.41	1.34	5.16	3.47
Kok and Schön [11]	0.71	1.79	1.38	6.65	2.28	1.24	1.34	4.81	4.82
Vasconcelos et al. [1]	**0.71**	12.79	13.68	17.95	1.81	1.24	1.34	5.16	3.47
MAGICAL [7]	0.71	1.00	15.48	27.76	1.35	1.14	1.34	2.78	2.52
L_1-norm ellipsoid fitting (SDP)	0.75	**0.71**	0.76	1.47	0.77	0.80	**1.34**	1.41	1.39
Autocal (L_1, **ours**)	0.72	0.82	1.01	0.81	0.77	0.79	1.34	1.40	1.37
Autocal (Cauchy, **ours**)	0.72	0.79	0.81	0.78	0.75	0.75	1.34	1.43	**1.34**
Autocal (Geman-McClure, **ours**)	0.73	0.84	**0.73**	0.80	**0.73**	**0.72**	1.35	1.47	1.36
Autocal (Huber, **ours**)	0.71	0.78	1.19	**0.78**	0.80	0.82	1.34	**1.39**	1.37

1) Data Collection: We set up an IMU sensor (InvenSense™ MPU 9250) and obtained raw data via I^2C communication using an arduino microprocessor. The sensor board was moved in arbitrary directions to obtain a set (track) of magnetometer readings.

We changed the surrounding environment to produce measurement tracks with and without outliers. First, we obtained two data tracks, M1 and M2, each of which comprises inlier data points only. Second, we retrieved two sets of sensor readings, MP1 and MP2, moving toward and away from a mobile phone to incur magnetic disruptions for inducing anomalies. Finally, we took three measurement sequences near a stapler, denoted as MS1–MS3 respectively. These also contain outliers due to arbitrary movements around a ferromagnetic object.

We also fetched accelerometer readings to demonstrate our algorithm's applicability in self-calibration of three-axis accelerometers. Again, we obtained two inlier-only data tracks, A1 and A2, and two outlier-inclusive tracks, AO1 and AO2, triggered by sudden accelerations during sensor movements. As our accelerometer readings are about 100 times bigger than those of the magnetometer, we have divided accelerometer values by 100 (assuming similar signal-to-noise ratio).

More details about each sequence can be found in Table III.

2) Experimental Procedure: As it is difficult to obtain ground truths for real data, we compared two quantities, namely, the normalized model deviation error (the same as the metric used in Section III-B3) and average sensor estimation error for checking algorithm precision and accuracy. Both metrics are explained in the following.

1) *Normalized Model Deviation Error (Precision):* We obtained each algorithm's reference model ($\mathtt{K}_{\text{ref}}$, $\mathbf{b}_{\text{ref}}$) by running it on an inlier-only data sequence M1. Then, its sensor model estimates on other data sets were compared against the reference by computing the normalized model error (also defined in (39)) for each data set as

$$\frac{\|[\mathtt{K} \quad \mathbf{b}] - [\mathtt{K}_{\text{ref}} \quad \mathbf{b}_{\text{ref}}]\|_F}{\|[\mathtt{K}_{\text{ref}} \quad \mathbf{b}_{\text{ref}}]\|_F}. \tag{40}$$

Above measures how deviated each model estimate (from potentially outlier-existing data) is from the inlier-only reference data, implying algorithm precision.

2) *Average Sensor Estimation Error (Accuracy):* For each algorithm on each data set, we estimated the corresponding sensor model ($\mathtt{K}$, $\mathbf{b}$) and checked how well it fits the inlier-only sequence M1 by comparing the the average sensor estimation error. For this purpose, we first derived a set of 3-D field directions minimizing the sum of squared sensor estimation errors on M1, i.e., find field vectors solving

$$\underset{\{\hat{\mathbf{x}}_j\}}{\arg\min} \sum_{j=1}^{N} \|\mathtt{K}\hat{\mathbf{x}}_j + \mathbf{b} - \mathbf{m}_j\|_2^2 \tag{41}$$

given a model ($\mathtt{K}, \mathbf{b}$). (No robust kernel is included as M1 is outlier-free.) Then, the average sensor estimation error is computed as $((1/N)\sum_{j=1}^{N} \|\mathtt{K}\hat{\mathbf{x}}_j + \mathbf{b} - \mathbf{m}_j\|_2^2)^{1/2}$. In terms of model learning, this can be viewed as computing the test accuracy of each algorithm after training the sensor model.

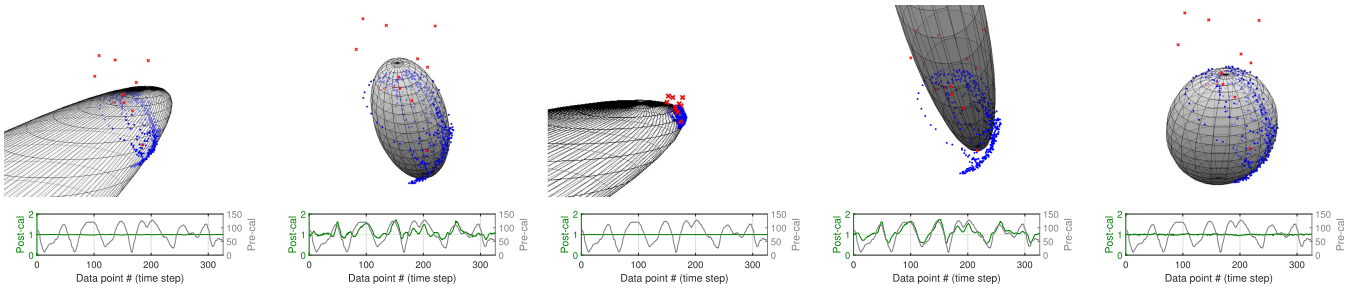


Fig. 7. Visualizations of the sensor models outputted by individual algorithms on the MS1 sequence from Table III. The top row shows each algorithm's ellipsoid model along with the sensor measurements, where 442 blue dots denote inlier data points and 12 red crosses are anomalies. The L_2-norm-based methods [see Fig. 7(a)–(d)] try their best to fit an ellipsoid across both inliers and outliers, leading to skewed models. On the other hand, our algorithm in Fig. 7(e) utilizes a robust kernel to downweight outliers, yielding a solution that is closer to the model obtained with inlier data points only. The bottom row illustrates the precalibration/postcalibration performance of the tested algorithms by plotting field size per data point when the top row's models are tested on the outlier-free M1 data set. These show a flat green curve close to 1 does not necessarily imply good calibration, and thus, one should always consult the ellipsoid fitting results. (a) Dorveaux *et al.* (b) Kok and Schön. (c) Vasconcelos *et al.* (d) MAGICAL. (e) Ours (Cauchy).

3) Results on Real Data: Normalized model deviation errors and average sensor estimation errors explained in Section III-B2 can be found in Tables IV and V, respectively. Numbers in bold represent the winning algorithms.

On the inlier only data sets (M2 and A2), the differences in accuracy and precision between our family of algorithms and state-of-the-art L_2-based method (MAGICAL [7]) are comparatively small. This shows that our method can yield accurate model values when no or few outliers are present.

For the outlier-existing sequences (MP1, MP2, MS1, MS2, MS3, AO1, and AO2), our Autocal family of algorithms performs substantially better in terms of both accuracy and precision than other L_2-norm minimizing methods from Table I. Our GM kernel implementation wins on most of the sequences, but its performance does vary from track to track, and we confirm the simulation result that the kernel is sensitive to the distribution of measurement noise. While the Cauchy kernel version only wins on AO2, its variation is smaller than other Autocal variants, producing more consistent models across all data tracks and thus recommended.

On a minor note, it is interesting to observe in Table IV that the algorithms of Dorveaux *et al.* and Vasconcelos *et al.* and MAGICAL sometimes produce completely inaccurate results. This is more severe for Vasconcelos *et al.*'s method in MP1, MP2, and MS1. To look for the cause, we have plotted resultant ellipsoids from the sensor models $(\mathtt{K}, \mathbf{b})$ outputted by individual algorithms in Figs. 1 and 7. This shows that the L_2-norm minimizing algorithms overall yield skewed and badly scaled ellipsoids for the real tracks tested in Table III. We have found that this is due to several of these tracks comprising inlier points mostly only on one side of the ellipsoid, encouraging L_2-based algorithms to yield a long and skewed ellipsoid in an attempt to fit both inliers and outliers at the same time.

We investigated further on the above issue by running each L_2-norm minimizing algorithm on a combined track comprising M1 and MP1, which covers more surface area of the ellipsoid. All but Dorveaux *et al.*'s method have reached feasible solutions with less than 1.5% normalized deviation and around 0.70 mean sensor estimation error. (Further analysis on Dorveaux *et al.*'s method showed that its failure on real data is due to poor initialization, and a good initial solution provided by L_1 ellipsoid fitting yields a better result.) This introduces another need for robust autocalibration in practice as the self-calibration motion may not be completely arbitrary, inflicting a detrimental impact on the L_2-norm-based methods.

TABLE VI
AVERAGE ALGORITHM RUNTIMES ON REAL MAGNETOMETER DATA SETS (MP1, MP2, MS1, MS2, AND MS3)

Algorithm	Mean runtime (s)
L_2-norm (least squares) ellipsoid fitting	0.002
Dorveaux et al. [8]	0.035
Kok and Schön [11]	3.891
Vasconcelos et al. [1]	0.014
MAGICAL (Papafotis and Sotiriadis) [7]	0.032
L_1-norm ellipsoid fitting (Sec. II-C)	0.177
Autocal (L_1, **ours**)	1.086
Autocal (Cauchy, **ours**)	0.303
Autocal (GM, **ours**)	0.353
Autocal (Huber, **ours**)	0.297

As a final remark, Fig. 7 shows incorrect badly scaled models are often not in the postcalibration plots (field size versus time step) produced by these skewed models [see Fig. 7(b) and (c)]. This is because the size of the field obtained by computing $\mathtt{K}^{-1}(\mathbf{m}_j - \mathbf{b})$ can always be near 1 if the badly scaled ellipsoid triggers two conditions, $\|\mathtt{K}^{-1}\mathbf{m}_j\|_2 \ll \|\mathtt{K}^{-1}\mathbf{b}\|_2$ and $\|\mathtt{K}^{-1}\mathbf{b}\|_2 = 1$. Hence, one should always prioritize the ellipsoid visualization for model validation.

4) Algorithm Runtimes: The average runtimes of individual algorithms are reported in Table VI. Since our implementation has not exploited all the efficient techniques available, we believe that there is a room for further improvement in runtime.

Despite higher accuracy and precision of our method, the family of proposed algorithms is slower than other L_2-norm-based algorithms except for Kok and Schön's method. The Cauchy, GM, and Huber kernels are faster than L_1. This is potentially due to the ill-conditioned nature of the L_1 kernel having a large gradient near-zero error preventing early convergence.

We have found that Calafiore's convex L_1-norm ellipsoid fitting algorithm [12] can benefit from MATLAB vectorization, yielding a factor of ten speedup and decreasing our method's runtime. Nevertheless, it still requires more computation power

than most L_2-norm minimizing methods. Improving the speed of robust ellipsoid fitting is for future work.

IV. Conclusion

In this work, we have proposed a robust autocalibration method that can be applied to a conventional three-axis magnetometer and accelerometer. The method works in two steps, initially estimating sensor variables through L_1-norm ellipsoid fitting of measurement data, followed by refinement of model variables via robust nonlinear least-squares minimization of sensor estimation errors. We have demonstrated that our method achieves higher calibration accuracy and precision in the presence of magnetic disturbances. This shows that consistent robust handling of outlier data as well as minimizing sensor estimation error is required to improve calibration accuracies in simulation and in practice. We hope future work would focus on devising a simpler and more efficient robust self-calibration algorithm for magnetometers without compromising accuracy.

References

[1] J. F. Vasconcelos, G. Elkaim, C. Silvestre, P. Oliveira, and B. Cardeira, "Geometric approach to strapdown magnetometer calibration in sensor frame," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, no. 2, pp. 1293–1306, Apr. 2011.

[2] H. K. Kuga and V. Carrara, "Attitude determination with magnetometers and accelerometers to use in satellite simulator," *Math. Problems Eng.*, vol. 2013, pp. 1–6, Aug. 2013.

[3] I. Vallivaara, J. Haverinen, A. Kemppainen, and J. Roning, "Simultaneous localization and mapping using ambient magnetic field," in *Proc. IEEE Conf. Multisensor Fusion Integr.*, Sep. 2010, pp. 14–19.

[4] H. Zhang and F. Martin, "Robotic mapping assisted by local magnetic field anomalies," in *Proc. IEEE Conf. Technol. Practical Robot Appl.*, Apr. 2011, pp. 25–30.

[5] A. Kemppainen, I. Vallivaara, and J. Roning, "Magnetic field SLAM exploration: Frequency domain Gaussian processes and informative route planning," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, Sep. 2015, pp. 1–7.

[6] K. N. Choi, "Metal detection sensor utilizing magneto-impedance magnetometer," *J. Sensors*, vol. 2018, pp. 1–9, Jun. 2018.

[7] K. Papafotis and P. P. Sotiriadis, "MAG.I.C.AL.—A unified methodology for magnetic and inertial sensors calibration and alignment," *IEEE Sensors J.*, vol. 19, no. 18, pp. 8241–8251, Sep. 2019.

[8] E. Dorveaux, D. Vissiere, A.-P. Martin, and N. Petit, "Iterative calibration method for inertial and magnetic sensors," in *Proc. 48h IEEE Conf. Decis. Control (CDC) Held Jointly 28th Chin. Control Conf.*, Dec. 2009, pp. 8296–8303.

[9] M. Kok and T. B. Schön, "Maximum likelihood calibration of a magnetometer using inertial sensors," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 92–97, 2014.

[10] Y. Wu and W. Shi, "On calibration of three-axis magnetometer," *IEEE Sensors J.*, vol. 15, no. 11, pp. 6424–6431, Nov. 2015.

[11] M. Kok and T. B. Schon, "Magnetometer calibration using inertial sensors," *IEEE Sensors J.*, vol. 16, no. 14, pp. 5679–5689, Jul. 2016.

[12] G. Calafiore, "Approximation of n-dimensional data using spherical and ellipsoidal primitives," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 32, no. 2, pp. 269–278, Mar. 2002.

[13] C. C. Foster and G. H. Elkaim, "Extension of a two-step calibration methodology to include nonorthogonal sensor axes," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 3, pp. 1070–1078, Jul. 2008.

[14] D. Gebre-Egziabher, G. H. Elkaim, J. D. Powell, and B. W. Parkinson, "Calibration of strapdown magnetometers in magnetic field domain," *J. Aerosp. Eng.*, vol. 19, no. 2, pp. 87–102, Apr. 2006.

[15] K. Kloster. (2014). *Matrix Computations*. [Online]. Available: math.purdue.edu/~kkloste/cs515fa14/qr-uniqueness.pdf

[16] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2008.

[17] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[18] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optim. Methods Softw.*, vol. 11, nos. 1–4, pp. 625–653, Jan. 1999.

[19] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Rev.*, vol. 38, no. 1, pp. 49–95, 1996.

[20] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.

[21] A. Fusiello, "Elements of geometric computer vision," Tech. Rep., 2006.

[22] S. Agarwal, K. Mierle, and Others. *Ceres Solver*. Accessed: Jul. 2020. [Online]. Available: ceres-solver.org

[23] C. Zach and G. Bourmaud, "Iterated lifting for robust cost optimization," in *Proc. Brit. Mach. Vis. Conf.*, Sep. 2017, p. 86.

[24] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quart. Appl. Math.*, vol. 2, no. 2, pp. 164–168, 1944.

[25] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, Jun. 1963.

[26] T. P. Minka, "Old and new matrix algebra useful for statistics," Microsoft Res., Redmond, WA, USA, Tech. Rep., 2000. [Online]. Available: tminka.github.io/papers/matrix/minka-matrix.pdf

[27] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—A modern synthesis," in *Proc. Int. Workshop Vis. Algorithms, Theory Pract.*,. Berlin, Germany: Springe, 2000, pp. 298–372.

[28] T. Okatani, T. Yoshida, and K. Deguchi, "Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 842–849.

[29] M. Grant and S. Boyd. (Mar. 2014). *CVX: MATLAB Software for Disciplined Convex Programming, Version 2.1*. [Online]. Available: http://cvxr.com/cvx

Je Hyeong Hong (Member, IEEE) received the M.Eng. degree in electrical and information sciences and the Ph.D. degree in engineering from the University of Cambridge, Cambridge, U.K., in 2011 and 2018.

He is currently serving military-replacement service as a Post-Doctoral Researcher at the Korea Institute of Science and Technology (KIST), Seoul, South Korea. His research interests include optimization problems and methods in sensors, electronics, and computer vision.

Donghoon Kang (Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from the Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 1997 and 1999, respectively, and the Ph.D. degree in mechanical and aerospace engineering from Seoul National University, Seoul, South Korea, in 2018.

Since 2000, he has been with the Korea Institute of Science and Technology (KIST), Seoul, where he is currently a Senior Researcher. He is also an Associate Professor with the KIST School, University of Science and Technology (UST), Daejeon, South Korea. His research interests include computer vision and machine intelligence.

Ig-Jae Kim (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Yonsei University, Seoul, South Korea, in 1996 and 1998, respectively, and the Ph.D. degree in electrical engineering and computer science from Seoul National University, Seoul, in 2009.

He was with the Massachusetts Institute of Technology (MIT) Media Lab, Cambridge, MA, USA, as a Post-Doctoral Researcher from 2009 to 2010. He is currently the Director-General of the Artificial Intelligence and Robotics Institute, Korea Institute of Science and Technology (KIST), Seoul. He is also an Associate Professor with the KIST School, University of Science and Technology (UST), Daejeon, South Korea, and a Guest Professor with Korea University. He has published over 80 fully referred papers in international journals and conferences, including the *ACM Transactions on Graphics*, SIGGRAPH, *Pattern Recognition*, and *Expert Systems With Applications*. His research interests include pattern recognition, computer vision, computer graphics, computational photography, and deep learning.