Received: 14 August 2018 | Revised: 18 April 2019 | Accepted: 20 June 2019

DOI: 10.1002/rob.21893

WILEY

REGULAR ARTICLE

Automatic targetless camera–LIDAR calibration by aligning edge with Gaussian mixture model

Jaehyeon Kang | Nakju L. Doh

Department of Electrical Engineering, Korea University, Seoul, South Korea

Correspondence
Nakju L. Doh, Department of Electrical Engineering, Korea University, Seong-buk, Seoul 136-713, South Korea.
Email: nakju@korea.ac.kr

Funding information
National Research Foundation of Korea, Grant/Award Number: 2011- 0031648; Korea Evaluation Institute of Industrial Technology, Grant/Award Number: 10073166; Korea Agency for Infrastructure Technology Advancement, Grant/Award Number: 18NSIP-B135768-02

Abstract

This paper presents a calibration algorithm that does not require an artificial target object to precisely estimate a rigid-body transformation between a camera and a light detection and ranging (LIDAR) sensor. The proposed algorithm estimates calibration parameters by minimizing a cost function that evaluates the edge alignment between two sensor measurements. In particular, the proposed cost function is constructed using a projection model-based *many-to-many* correspondence of the edges to fully exploit measurements with different densities (dense photometry and sparse geometry). The alignment of the many-to-many correspondence is represented using the Gaussian mixture model (GMM) framework. Here, each component of the GMM, including weight, displacement, and standard deviation, is derived to suitably capture the intensity, location, and influential range of the edge measurements, respectively. The derived cost function is optimized by the gradient descent method with an analytical derivative. A coarse-to-fine scheme is also applied by gradually decreasing the standard deviation of the GMM to enhance the robustness of the algorithm. Extensive indoor and outdoor experiments validate the claim that the proposed GMM strategy improves the performance of the proposed algorithm. The experimental results also show that the proposed algorithm outperforms previous methods in terms of precision and accuracy by providing calibration parameters of standard deviations less than 0.6° and 2.1 cm with a reprojection error of 1.78 for a 2.1-megapixel image (2,048 × 1,024) in the best case.

KEYWORDS
calibration, perception, sensor networks, sensors

1 | INTRODUCTION

Cameras and light detection and ranging (LIDAR) sensors are widely employed sensors in many robotics applications, such as autonomous driving (Wei et al., 2013), classification (Douillard, Fox, Ramos, & Durrant-Whyte, 2011), segmentation (Strom, Richardson, & Olson, 2010), object detection (González, Vázquez, López, & Amores, 2017), and mapping (J. Zhang & Singh, 2015). This widespread use stems from the fact that the dense photometric measurement of a camera and LIDAR's sparse geometric measurement can assist each other. To take advantage of the complementary measurements of a camera and LIDAR, we need to precisely calibrate the six-degree-of-freedom (6-DOF) rigid-body transformation between the two sensors.

Early calibration methods mainly made use of an artificial target object observable by both sensors, such as a checkerboard (Geiger, Moosmann, Car, & Schuster, 2012; Mirzaei, Kottas, & Roumeliotis, 2012; Unnikrishnan & Hebert, 2005; Zhang & Pless, 2004). However, the data acquisition of a target-based method is laborious and time-consuming. Additionally, the target object limits the applicability of the algorithms, because the calibration data must be obtained in a controlled environment.

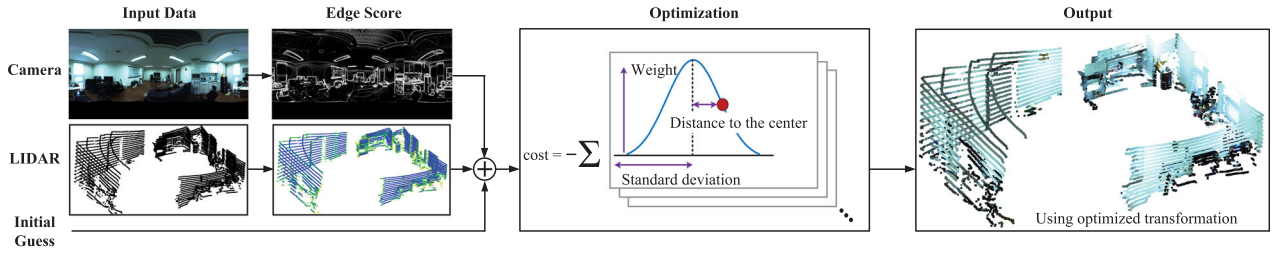

FIGURE 1 Overview of the proposed algorithm. First, for the preprocessing, we extract the edge scores of each pixel and each point in the image and the point cloud. Next, we calculate the Gaussian mixture model variables (weight, distance to the center, and standard deviation) by combining the extracted edges and the corresponding edge scores with the current transformation parameters. The proposed method optimizes the cost function, which is calculated using the acquired Gaussian mixture model variables to estimate the accurate transformation parameters between the camera and the LIDAR [Color figure can be viewed at wileyonlinelibrary.com]

Recently, more general algorithms have been developed to solve the limitations of target-based methods by calibrating sensors *without* using an artificial target object. Targetless methods expand the usability of the calibration algorithm to various applications, such as the detection of calibration drift over time (Levinson & Thrun, 2013) or camera localization in a geometry map (Pascoe, Maddern, & Newman, 2015). These advances arise from the generic nature of target-free input measurements, which allows the calibration framework to be performed in an uncontrolled environment. Targetless approaches in an uncontrolled environment also require less effort to obtain calibration data.

Although a target object is no longer required, the underlying structure of targetless calibration methods, which is equivalent to that of target-based methods, can be outlined in two steps. The first step involves (i) the definition of a cost (or score) function with the sensor measurements and the second step involves (ii) its optimization.

When (i) defining a cost function, most of the previous works employed a *one-to-one* correspondence between the measurements of the camera and LIDAR. However, this approach uses only a part of the camera's dense photometric measurement by discarding the pixels that do not match LIDAR's sparse geometric measurement. On the contrary, a cost function based on a *many-to-many* correspondence can fully exploit both measurements, regardless of their density differences (Agamennoni, Fontana, Siegwart, & Sorrenti, 2016) and improve the performance of the algorithm. Regarding (ii) the optimization step, the calculation of an *analytical gradient* is more accurate than a *numerical approximation*, which is the approach of most previous works.

In this paper, we present a many-to-many correspondence-based metric, which is analytically differentiable and which does not require a target to calibrate the camera-LIDAR sensor system. The proposed method estimates the calibration parameters by aligning the edges of both sensor measurements, as in Figure 1. Here, we define a cost function under the assumption that edge pixels and projections of edge points form similar distributions on the image if we project edge points with the correct calibration parameters.

More explicitly, we project the edge point of LIDAR onto the image. Then, the Gaussian kernel filters the distance between each worthwhile pair[1] of edge point projections and edge pixels. The weight of each pair is defined to reflect the "edginess" of the point and the pixel. We also define the standard deviation of the Gaussian kernel by considering the projection model. Given the determined values of distance, weight, and standard deviation, we represent the overall calibration cost as a weighted sum of Gaussian functions, which is the Gaussian mixture model (GMM). Specifically, the contributions of this paper are as follows:

- An introduction of the GMM calibration framework that systematically combines a many-to-many correspondence of edge measurements while considering the projection model.
- An analytical gradient calculation of the proposed cost function to enhance the estimation accuracy of the optimization.
- A proposal of an edge score metric, which increases as a query point approaches the boundary edge or the fold edge of a point cloud.
- An experimental validation of the proposed algorithm and a comparison with other algorithms.

The rest of this paper is organized as follows: Section 2 reviews previous targetless camera-LIDAR calibration algorithms. The modeling equation of the sensor system is then presented in Section 3. The edge scores representing the "edginess" of each measurement are derived in Section 4 and used in formulating the proposed cost function in Section 5. Additionally, the proposed cost function is optimized, as described in Section 6. The calibration method proposed in Section 7 is then experimentally validated in Section 8. Section 9 provides discussions of the proposed method and then Section 10 concludes that the proposed GMM framework precisely calibrates the camera-LIDAR sensor system.

2 | RELATED WORK

This section focuses on examining targetless calibration methods as most recent works, including this study, attempt to calibrate without

[1]A pair representing the distance within three-sigma of the Gaussian.

an artificial target. Specifically, we organize the previous works from the correspondence-matching perspective.

One way to reduce the need for a target object is to create a *synthetic image* from LIDAR measurements and compare the "*pixel-to-pixel*" correspondence to a camera image. For this purpose, Napier, Corke, and Newman (2013) generate a synthetic image from a prior map, with its intensity as a reflection value of LIDAR. Then, the weighted sum of squared differences (SSD), using the intensities of the camera image and the synthetic image, is minimized by a brute force method.

The intensities of a camera image and a synthetic image can also be matched by probabilistic metrics (Pascoe et al., 2015; Taylor & Nieto, 2012; Wang, Ferrie, & Macfarlane, 2012). These metrics are functions of the marginal and the joint probabilities, where the joint probability is acquired from the pixel-to-pixel correspondence.

Regarding probabilistic metrics, Taylor and Nieto (2012) generate a synthetic LIDAR image, with its intensity represented by the angle of a point's normal vector. Then, they compare the intensity of the real and synthetic images by using the sum of marginal entropies divided by the joint entropy. Lastly, a particle swarm algorithm optimizes the calculated metric to acquire the calibration parameters. Additionally, Wang et al. (2012) represent the intensity of a synthetic image by the LIDAR's intensity or a depth map. To evaluate the misalignment, they employ mutual information (MI), which is the subtraction of the joint entropy from the sum of marginal entropies. Then, the Nelder–Mead simplex method optimizes it for the calibration.

These approaches measure the similarity between distributions as a function of the marginal and the joint probabilities. Thus, the authors calculate the intensity probabilities of the real and synthetic images by using a histogram, and the discrete nature of the histogram limits these methods to numerical optimization. To resolve this limitation, Pascoe et al. (2015) represent the histogram by a B-spline curve to derive the analytical gradient of the cost function. Then, the normalized inverse distance metric, which is also a function of marginal and joint probabilities, is optimized using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) quasi-Newton method (Nocedal & Wright, 1999).

Nevertheless, the abovementioned synthetic image-based approaches have a limitation in that they require dense geometry information, such as a dense prior map (Napier et al., 2013; Pascoe et al., 2015; Wang et al., 2012) or dense LIDAR sensor (Taylor & Nieto, 2012), to generate a synthetic image.

Another way to calibrate a camera-LIDAR sensor system is to extract the *geometric feature* from both measurements and register the "*feature-to-feature*" correspondence. Scaramuzza, Harati, and Siegwart (2007) select points (1D features) from a camera image and a synthetic LIDAR image. Then, they find calibration parameters by using the perspective-n-points (PnP) algorithm. Moghadam, Bosse, and Zlot (2013) obtain lines (2D features) from an image and a point cloud. They project the line from the point cloud onto the image and find the two projected points that are closest to the endpoints of the line segment in the image; the distance between the points is then minimized by a numerical gradient. Tamas and Kato (2013) extract planar regions (3D features) in both measurements by using the region growing algorithm. Then, the numerical Levenberg–Marquardt algorithm optimizes the overlapping area of the regions.

When registering each geometric feature, it is essential to determine the feature-to-feature correspondence between an image and a point cloud. However, all previous approaches (Moghadam et al., 2013; Scaramuzza et al., 2007; Tamas & Kato, 2013) require a manual correspondence-selection stage to find the feature pair, which is only partially automatic and thus time-consuming.

Rather than converting the raw measurements into a synthetic image or a geometric feature, calibration parameters can also be determined by aligning *raw measurements* with the "*pixel-to-point*" correspondence. The fundamental idea of measurement-based approaches is that the pixels and points become highly correlated after the points are projected onto the image with correct calibration parameters.

Specifically, Pandey, McBride, Savarese, and Eustice (2015) project a point onto an image and combine the reflectivity of the LIDAR measurement and the image intensity to calculate the same MI metric as in Wang et al. (2012). Here, the joint probability is obtained from the one-to-one correspondence between the pixel and the projected point. In the optimization step, the Barzilai–Borwein steepest gradient-ascent algorithm with a numerical gradient maximizes the MI metric for the calibration.

Levinson and Thrun (2013) introduce the idea that the alignment of the edge correspondences between pixels and points can also be employed for calibration. To that end, they construct the edge image by smoothing the intensity difference between neighboring pixels, such that the intensity of the edge image gradually increases as a pixel approaches to an edge. Additionally, the edge points of LIDAR measurements are extracted using the range discontinuity, where the intensity increases as the discontinuity increases. Then they formulate a cost function by multiplying intensities of the pixel and the projected edge point correspondence. Finally, the cost is optimized using the grid search algorithm.

Similarly, Taylor, Nieto, and Johnson (2015) compare the gradients of correspondences between pixels and points to calibrate the camera and the LIDAR. The image gradient is calculated by conventional methods with a Gaussian blur, and the LIDAR gradient is computed after projecting points onto the image with their intensities. Instead of solely comparing edge intensity, they integrate the intensity and the orientation of gradients to calculate a cost function. The Nelder–Mead simplex method then minimizes the cost function to acquire the calibration parameters.

The method proposed in this paper also employs the pixel-to-point correspondence for the calibration. However, unlike existing methods, which use one-to-one correspondence or many-to-many correspondence by a uniform blurring,[2] we apply

[2]Previous methods, including Levinson and Thrun (2013) and Taylor et al. (2015), use one-to-one correspondence between a blurred image and a point cloud. Image blurring is equivalent to using the many-to-many correspondence between an original image and a point cloud, because multiple original pixels generate a pixel in the blurred image.

FIGURE 2 Sensor system consisting of a panoramic camera and a 3D LIDAR. The panoramic camera provides a dense 2D photometric measurement of the environment, whereas the 3D LIDAR measures the 3D sparse geometry. The sensor system acquires the colored point cloud by calibrating the 3D rigid-body transformation between the two sensors [Color figure can be viewed at wileyonlinelibrary.com]

a projection model-based many-to-many correspondence between pixels and points. In particular, we consider the projection characteristics influenced by the distance to a 3D point and a camera projection model. Thus, the proposed method adaptively constitutes the many-to-many correspondence to enhance the calibration performance. In the following sections, we present a novel GMM framework with the projection model-based many-to-many correspondence of pixels and points, which can be optimized by the analytical gradient.

3 | MODELING

In this study, we construct a sensor system with a panoramic camera (Ladybug5) and a 3D LIDAR (Velodyne VLP-16 LIDAR), as shown in Figure 2.

The measurements of each sensor are as follows:

- Camera measurements: dense intensity (grayscale) measurement, which composes a calibrated spherical panoramic image $I \in \mathbb{R}^{h\times w}$, where h and w are the height and the width of the image, respectively.
- LIDAR measurements: distance and corresponding horizontal and vertical angles of a ray, which sweeps an environment around the LIDAR. The measurements generate a sparse point cloud by stacking 3D points $\boldsymbol{p} \in \mathbb{R}^3$.

With the available measurements, the calibration problem is defined as an estimation of a 3D rigid-body transformation of the LIDAR coordinate frame relative to the camera coordinate frame (C). Here, we define both coordinate frames such that each sensor is facing the x-axis direction, and the z-axis points in the upward direction with the right-hand rule.

The rigid-body transformation that we aim to estimate is defined by a 3-DOF rotation and a 3-DOF translation as follows:

$$\boldsymbol{R} \in SO(3), \tag{1}$$

$$\boldsymbol{t} \in \mathbb{R}^3, \tag{2}$$

which transforms the i-th point $\boldsymbol{p}_i$ of the LIDAR measurements

$$\boldsymbol{p}_i^c = \boldsymbol{R}\cdot\boldsymbol{p}_i + \boldsymbol{t}, \tag{3}$$

to obtain the point with respect to the camera coordinate frame C.

In this study, we parameterize the rotation matrix $\boldsymbol{R}$ by using a vector $\boldsymbol{\omega}$ in $\mathfrak{so}(3)$, which is the tangent space to the 3D rotation group $SO(3)$. The vector $\boldsymbol{\omega}$ constructs the corresponding rotation matrix as follows:

$$\boldsymbol{R}(\boldsymbol{\omega}) = \exp(\boldsymbol{\omega}^\wedge), \tag{4}$$

and gives a derivative of the rotated point $\boldsymbol{R}(\boldsymbol{\omega})\cdot\boldsymbol{p}$, as in Section 6. Here, $\boldsymbol{\omega}^\wedge$ represents the skew-symmetric matrix of the vector $\boldsymbol{\omega}$.

It is also possible to represent both the rotation and the translation by a vector in $\mathfrak{se}(3)$ which is the tangent space to the special Euclidean group $SE(3)$. However, we maintain the translation vector in $\mathbb{R}^3$ for computational efficiency, as described in Manton (2002).

4 | EDGE SCORE

In this section, we explain the method of extracting edge scores from the image and the point cloud. Specifically, we compute edge scores that increase as the measurements approach the edge in each measurement space (2D image space for the camera and 3D Euclidean space for the LIDAR). The computed edge scores are used to extract the edge measurement and to calculate the weight in the proposed cost function.

4.1 | Image edge score

As in Section 3, we use a rectified grayscale image I with accurate intrinsic parameters provided by the manufacturer or the intrinsic calibration algorithm, as in Bouguet (2015).

For the rectified image, we smooth it by the L_0 gradient minimization method (Xu, Lu, Xu, & Jia, 2011). This smoothing step suppresses the details of an image by flattening small non-zero gradient regions while maintaining the salient edge pixels with a high gradient.

As a result, this image-smoothing reduces the noise in the image gradient, where its magnitude is used as the edge score $E_I(\boldsymbol{q})$ of a pixel $\boldsymbol{q}$. In this study, we calculate the image gradient by the

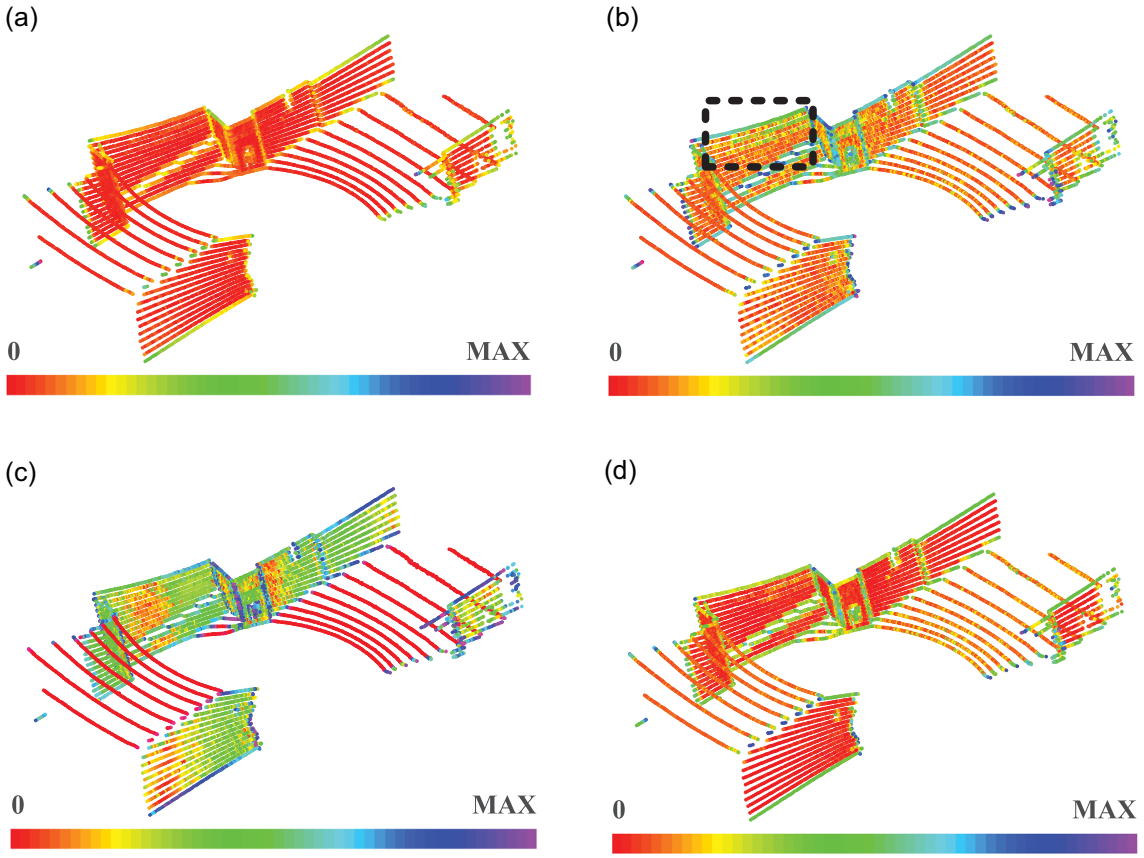

FIGURE 3 Metrics for detecting an edge score of a point cloud. Each point cloud is colored according to the (a) center-based score (Xia & Wang, 2017), (b) modified center-based score, (c) non-planarity, and (d) proposed edge score. Points within the radius of 0.1 m from the query point and the 30 nearest neighboring points are used to calculate each metric. As in (a), the center-based score shows indistinct results in edge regions. To the contrary, the modified center-based score in (b) presents improved distinction but also increased noise (false positive results), as in the region indicated by the black square. Thus, we define the proposed edge score by multiplying (b) the modified center-based score and (c) the non-planarity, which shows noticeable edge points with reduced noise, as in (d) [Color figure can be viewed at wileyonlinelibrary.com]

conventional Sobel operator (Sobel & Feldman, 1968) and then perform a non-maximum suppression of the image gradient for computational efficiency.

4.2 | Point cloud edge score

A point edge can be of two types (Ni, Lin, Ning, & Zhang, 2016). One type is a boundary edge, which comes from the geometric discontinuity at the border of an element in the environment. The other type is a fold edge due to a directional discontinuity at the intersection of distinct surfaces.

A boundary edge can be easily determined by a range discontinuity, as in Levinson and Thrun (2013), whereas the determination of a fold edge requires more complicated procedures. To simultaneously extract both types of edges, we modify the center-based score in Xia and Wang (2017) and combine it with a property of the local non-planarity as follows.

In Xia and Wang (2017), the authors derive the center-based score by using the nearest neighbor of a query point. The utilization of neighboring points is a natural way to extract edge points, because the edge property comes from the relationship between neighboring data. We can acquire the neighboring points in two ways. The first is the k-nearest neighbor algorithm, which finds k closest points to the query point. The second is the nearest neighbor algorithm with a fixed radius, which selects all points within a distance R from the query point.

The center-based score employs both the nearest neighbor algorithms to acquire the point set, which reliably represents the local environment, even with the various point cloud densities. Then, the center-based score is defined as a distance between the query point $\boldsymbol{p}$ and the center of the neighboring points, which is normalized by R. The score is higher for edge points, as shown in Figure 3a. However, the center-based score represents globally indistinct edge points (i.e., the edge points are not clearly separated), as shown in Figure 3a.

Rather than using the gradient of the center-based score as in Xia and Wang (2017),[3] we try to find an alternative globally distinct metric to extract edge points with a simple threshold. To that end, we modify the normalization factor R and define it to be the maximum distance from the neighboring points to the query point, thereby generating the modified center-based score $E_1(\boldsymbol{p})$ as follows:

$$E_1(\boldsymbol{p}) = \frac{1}{\max\left(d\left(\boldsymbol{p}, \mathcal{N}(\boldsymbol{p})\right)\right)} \left\| \boldsymbol{p} - \frac{1}{n} \sum_{i \in \mathcal{N}(\boldsymbol{p})} \boldsymbol{p}_i \right\|, \tag{5}$$

where $\mathcal{N}(\boldsymbol{p})$ represents the point set, which includes n neighboring points of the query point $\boldsymbol{p}$ by using both the nearest neighbor algorithms, and $d(\boldsymbol{p}, \mathcal{N}(\boldsymbol{p}))$ indicates a vector of distances between the query point and the neighboring points. The modified score of the point cloud is shown in Figure 3b. As in figure, the modified score shows improved distinction in the edge area. However, some non-edge points show noisy results, such as in the region indicated by the black square in the figure.

To compensate for the noisy points, we further employ a property of the local non-planarity under the assumption that locally planar points are less likely to be on the edge. Here, the local non-planarity $E_2(\boldsymbol{p})$ can be calculated as the difference between the local planarity (Weinmann, Jutzi, Hinz, & Mallet, 2015) and its maximum as follows:

$$E_2(\boldsymbol{p}) = 1 - \frac{\lambda_2 - \lambda_3}{\lambda_1}, \tag{6}$$

where λ_i ($\lambda_1 \geq \lambda_2 \geq \lambda_3$) indicates an eigenvalue of a covariance matrix constructed with the same neighboring points as in the computation of E_1. Then, the calculated non-planarity shows a small value on a planar region, as presented in Figure 3c.

Lastly, we define an edge score $E_P(\boldsymbol{p})$ of a point cloud by multiplying the modified center-based score and the non-planarity, as follows:

$$E_P(\boldsymbol{p}) = E_1(\boldsymbol{p}) \cdot E_2(\boldsymbol{p}), \tag{7}$$

with the query point $\boldsymbol{p}$. As $E_2(\boldsymbol{p})$ suppresses the noisy points of $E_1(\boldsymbol{p})$, the proposed edge score shows noticeable results at edge points (both boundary edges and fold edges). As a result, the proposed edge score generates more distinct differences between the edge points and the non-edge points, as shown in Figure 3d.

4.3 | Edge extraction

With the calculated edge scores $E_I(\boldsymbol{q})$ and $E_P(\boldsymbol{p})$, we extract the measurements that need to be aligned for the calibration.

In the proposed cost function, both edge scores are used in calculating the weight of a pixel-to-point correspondence (as in Section 5.1). The weight term makes a pixel-to-point correspondence with high edge scores influential in the overall cost. On the contrary, a pixel-to-point correspondence with low edge scores has a lesser effect on the overall cost. Therefore, we exclude the measurements of low edge scores by using thresholds on both edge scores for computational efficiency. In particular, when extracting edge points, we filter edge points from the top and bottom rays of the 3D LIDAR using the vertical angle of each point to remove false positives at the boundary of the scanning area, as in Figure 3d.

In determining the threshold values, it is allowed to extract the correct edges (high edge scores) and their nearby measurements (low edge scores) together with the roughly determined thresholds, because the weight term ensures that the extracted measurements mainly align with the correct edges. Accordingly, we can reduce the need for the difficult-to-select optimal thresholds for both edge scores.

Moreover, using the nearby measurements with low edge scores is similar to blurring the correct edges with high edge scores, which smooths out the proposed cost function. In other words, the nearby measurements increase the basin of convergence in an iterative optimization to avoid local minima.

5 | COST FUNCTION

In this section, we define the cost function to measure the geometrical difference between the projections of edge points and edge pixels under the assumption that edge pixels and edge points are in similar positions after projecting the edge points onto an image with the correct calibration parameters. Under this assumption, we calculate the displacement between the *many-to-many* correspondence of the edge pixels and the projection of the edge points. Then, the displacement is filtered by the negative Gaussian kernel, such that the proposed cost increases as the displacement increases, while providing continuous derivatives.

Intuitively, we can treat each projected edge point as a Gaussian center that independently provides a Gaussian distribution over the entire image, as in Figure 4. Then, we calculate the cost function according to the location of each edge pixel in the Gaussian distribution.

Specifically, the distance between the projection of the i-th edge point and the j-th edge pixel $d_{i,j}$ is filtered by the Gaussian kernel. Here, the distance $d_{i,j}$ that exceeds the three-sigma of the Gaussian distribution is excluded from the cost for computational efficiency, as its value is insignificant. The overall cost $GMM(\cdot)$ for the valid $d_{i,j}$ is then defined by the GMM, which consists of the weight and the standard deviation of each Gaussian distribution as follows:

$$GMM(\tau) = -\sum_{i=1}^{N_P} \sum_{j \in \Omega_i} w_{i,j} \cdot \boldsymbol{G}(d_{i,j}, \sigma_i), \tag{8}$$

where $\tau = [\boldsymbol{\omega}, \ \boldsymbol{t}] \in \mathbb{R}^{6\times 1}$ represents a vector indicating calibration parameters, as in Section 3, N_P is the number of edges in the point cloud, and Ω_i indicates the indices of the edge pixels within the three-sigma of the i-th Gaussian center. Note that the negative sign is applied to transform the calibration into a minimization problem.

[3] In Xia and Wang (2017), the authors detect the edge points by constructing the structure tensor of edge score changes. Then, the ratio between the trace and the determinant of the structure tensor is filtered with a threshold, as the Harris corner detection algorithm (C. Harris & Stephens, 1988) does in a 2D image.

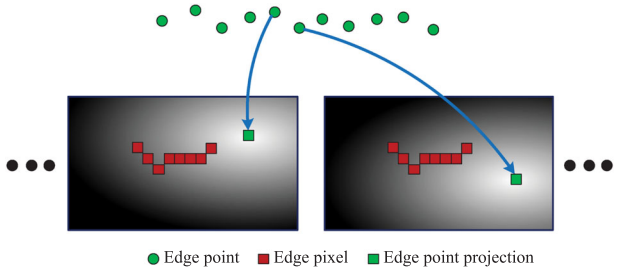

FIGURE 4 Illustration of the Gaussian distribution in the image. The green circles, red squares, and green squares indicate the edge points, edge pixels, and projections of the edge points, respectively. Each Gaussian distribution has its center at the projection of each edge point in the image. Then, we calculate the cost by using the position of each edge pixel in each Gaussian distribution [Color figure can be viewed at wileyonlinelibrary.com]

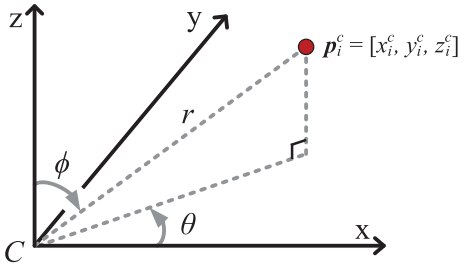

FIGURE 5 Conversion of Cartesian coordinates to spherical coordinates. A point $\boldsymbol{p}_i^c$ (red circle) in the Cartesian coordinate system is converted to a range (r), azimuth angle (θ), and inclination angle (ϕ) according to Equations (11)–(13). C indicates the center of the camera coordinate frame, where the camera faces in the direction of the x-axis [Color figure can be viewed at wileyonlinelibrary.com]

Additionally, we define the Gaussian kernel $\boldsymbol{G}(d_{i,j}, \sigma_i)$ of the equation, which is weighted by $w_{i,j}$ as follows:

$$\boldsymbol{G}(d_{i,j}, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{d_{i,j}^2}{2\sigma_i^2}\right). \tag{9}$$

Here, σ_i is the standard deviation derived to represent the influential range of the edge point by considering the projection model. The subsequent subsections derive each part of the cost function: the weight $w_{i,j}$, distance to the center $d_{i,j}$, and standard deviation σ_i.

5.1 | Weight calculation

The cost function constructed using the GMM has a weight term, as in (8). The weight term reflects the importance of the corresponding pair of an edge pixel and edge point. Thus, their edge scores, as in Section 4, can be used to indicate the importance of the pair.

We define the weight by adding the normalized edge scores of both measurements (edge pixel $\boldsymbol{q}_j$ and edge point $\boldsymbol{p}_i$) as follows:

$$w_{i,j} = \frac{1}{2\cdot|\Omega_i|}\left(\frac{E_I(\boldsymbol{q}_j)}{\max(E_I)} + \frac{E_P(\boldsymbol{p}_i)}{\max(E_P)}\right), \tag{10}$$

where $|\Omega_i|$ is another normalizer indicating the number of edge pixels in Ω_i.

The normalization by the maximum value of each score adjusts the effects of the edge scores E_I and E_P on the weight, such that it is not biased toward either measurement. Additionally, the normalization by the number of edge pixels $|\Omega_i|$ suppresses the additive influence of the edge point $\boldsymbol{p}_i$ in the overall cost, which increases as the number of corresponding edge pixels increases and, in return, limits the maximum cost due to the edge point being less than one.

5.2 | Displacement calculation

The displacement in (8) is a geometric difference between an edge pixel and the projection of an edge point in the image space. Before taking the projection, an edge point needs to be transformed to the camera coordinate frame, as in (3), with the calibration parameters.

We then convert the transformed point $\boldsymbol{p}_i^c = [x_i^c,\ y_i^c,\ z_i^c]$ to spherical coordinates as follows:

$$r(\boldsymbol{p}_i^c) = \sqrt{(x_i^c)^2 + (y_i^c)^2 + (z_i^c)^2}, \tag{11}$$

$$\theta(\boldsymbol{p}_i^c) = \text{atan2}(y_i^c, x_i^c) \in [-\pi, \pi], \tag{12}$$

$$\phi(\boldsymbol{p}_i^c) = \text{acos}\left(\frac{z_i^c}{r}\right) \in [0, \pi], \tag{13}$$

where r is the distance to the point from the origin of the camera coordinate frame and θ and ϕ represent the azimuth and inclination angles, respectively, as in Figure 5.

Then, the edge point $\boldsymbol{p}_i$ is projected onto the pixel $\boldsymbol{q}_i = [u_i,\ v_i]$ as follows:

$$\boldsymbol{q}(\boldsymbol{p}_i^c) = [u(\boldsymbol{p}_i^c),\ v(\boldsymbol{p}_i^c)], \tag{14}$$

$$u(\boldsymbol{p}_i^c) = w\cdot\frac{\pi - \theta(\boldsymbol{p}_i^c)}{2\pi} \in [0, w], \tag{15}$$

$$v(\boldsymbol{p}_i^c) = h\cdot\frac{\phi(\boldsymbol{p}_i^c)}{\pi} \in [0, h], \tag{16}$$

where w and h represent the width and height of the image, and $u(\cdot)$ and $v(\cdot)$ are the coordinates of the projected edge point along the u-axis and v-axis of the image coordinate system.

We then calculate the displacement as the shortest distance between the edge pixel and the projected edge point. As in Figure 6, when moving along horizontal and vertical axes, a spherical panoramic image has four routes between two points in the image. These routes originate from the periodicity of angles, as represented by the horizontal and vertical circles on the sphere in Figure 6.

To obtain the displacement between the two points, we first calculate the length of each route between the two points as the L2-norm of the traversed lengths in the horizontal and vertical directions. The minimum of the four traversed lengths is then

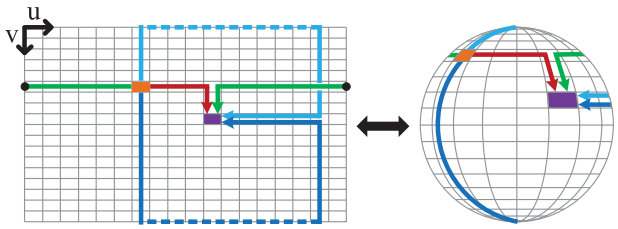

FIGURE 6 Four routes from the starting point (orange) to the goal point (purple) of a spherical panoramic image. The same routes are also depicted on the sphere. Of the four routes, the route with the minimum distance is selected to determine a displacement between the two points. The distance is calculated as the L2-norm of traversed lengths along the *u*- and the *v*-axes of the image. Ordinarily, the trivial route (red or green arrow) indicates the minimum distance. However, when the starting point or target point is near the top or bottom of the image, other routes can also have the minimum distance [Color figure can be viewed at wileyonlinelibrary.com]

FIGURE 7 Pixel displacements in a spherical panoramic image represented by the same geometric displacement in 3D Euclidean space. The lights in the image are scattered at a constant distance in the actual environment. Each vector (arrow) roughly indicates the pixel displacement between lights, while vectors of the same color represent the same geometric displacement in 3D space. As in the figure, the same geometric displacement is represented by vectors with various lengths in the image [Color figure can be viewed at wileyonlinelibrary.com]

selected as the displacement between the projection of the edge point and the edge pixel.

5.3 | Deviation calculation

The goal of a camera-LIDAR calibration is to accurately apply the color data to the point cloud in 3D Euclidean space, whereas the proposed cost is calculated based on the displacement in the 2D image. The problem is that the displacement in the 2D image is not proportional to the geometric displacement in 3D Euclidean space, owing to the projection model. That is, the number of pixels required to represent the same geometric displacement is different, as shown in Figure 7. Owing to this *non-proportional displacement conversion*, each Gaussian distribution needs to have a different influential range to produce an accurately colored point cloud in 3D Euclidean space.

The standard deviation in (8) can be used for this purpose. The standard deviation controls the coverage of a Gaussian distribution in the image, such that a Gaussian distribution with a higher standard deviation covers a wider area. In other words, we need to increase the standard deviation in the regions that require more pixels to represent the same geometry in the Euclidean space.[4] To that end, we adjust the standard deviation of each Gaussian distribution by considering the projection characteristics including (i) the distance to the 3D point and (ii) the camera projection model as follows.

First, we discuss the distance to the 3D point. Owing to the perspective nature, an object becomes smaller in the image as the distance from the camera to the object increases. To capture this property, we use an edge point $\boldsymbol{p}_i$ whose projection to the image becomes the center of the Gaussian distribution. Specifically, we adjust the standard deviation σ_i to make it inversely proportional to the distance to the point as follows:

$$\sigma_i \propto \frac{1}{r(\boldsymbol{p}_i^c)}, \tag{17}$$

where r can be calculated as in (11) with the edge point $\boldsymbol{p}_i$.

Next, we analyze the effects of the camera projection model. The camera projection model shows how to project the real environment in 3D space onto the 2D image. Thus, the camera projection model naturally affects the non-proportional displacement conversion. In this study, we capture the effect of the camera projection model by examining the area of the region projected onto each pixel.

The spherical camera model, as discussed in Section 5.2, projects the environment onto the unit sphere and represents it by evenly distributed azimuth and inclination angles. In other words, the surface of the unit sphere is divided by these angles, as in Figure 8, and is projected onto a pixel $\boldsymbol{q} = [u,\ v]$. As in the figure, the unit sphere is first horizontally divided into h spherical segments, where its surface area is calculated as $2\pi k$ according to J. W. Harris and Stöcker (1998). Here, the height of a spherical segment k is acquired using the inclination angles ϕ^+ and ϕ^- to the upper and lower circles of the spherical segment as follows:

$$k(v) = \cos(\phi^+(v)) - \cos(\phi^-(v)). \tag{18}$$

Moreover, the inclination angles can be calculated by the inverse of (16) with the upper and lower sides of the square pixel ($v \pm 0.5$) as follows:

$$\phi^+(v) = (v + 0.5)\cdot\frac{\pi}{h}, \tag{19}$$

[4]If we increase the standard deviation, the Gaussian gives a less peaky value at its center which could disturb the convergence to it. Nevertheless, this is acceptable because we assign large standard deviations to areas where the pixel error exhibits a relatively small geometric error compared to the case of small standard deviations.

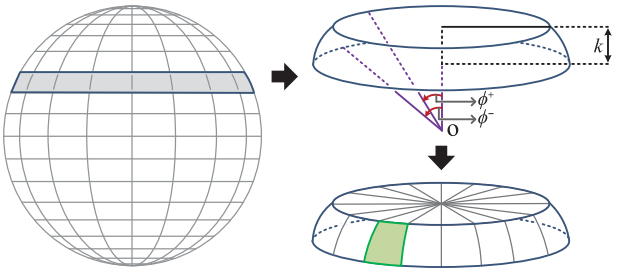

FIGURE 8 Procedure to extract the region projected onto each pixel with a spherical camera model. First, the unit sphere is sliced into h (height of the image) spherical segments by the equally divided inclination angles with respect to the center of the sphere (O). k indicates the height of the spherical segment and ϕ^+ and ϕ^- represent the inclination angles to upper and lower circles of the spherical segment, respectively. Next, the azimuth angles divide each spherical segment into w (width of the image) surfaces, which is the region (the green surface) projected onto each pixel [Color figure can be viewed at wileyonlinelibrary.com]

$$\phi^-(v) = (v - 0.5)\cdot\frac{\pi}{h}. \tag{20}$$

Then, the azimuth angles equally divide each spherical segment into w surfaces, as in Figure 8. Thus, the area S of the surface, which is projected onto each pixel, is given by

$$\begin{aligned} S(v) &= \frac{2\pi\cdot k(v)}{w} \\ &= \frac{2\pi}{w}\cdot(\cos(\phi^+(v)) - \cos(\phi^-(v))). \end{aligned} \tag{21}$$

If the area is large, we need fewer pixels to represent the same geometry. Thus, we adjust the standard deviation σ_i of the Gaussian distribution, centered at the projection of the edge point $\boldsymbol{p}_i$, to make it inversely proportional to the area of the region projected to the center pixel[5] as follows:

$$\sigma_i \propto \frac{1}{S(v(\boldsymbol{p}_i^c))}. \tag{22}$$

We then define the standard deviation σ_i of the Gaussian distribution by combining (17) and (22):

$$\sigma_i = \sigma_{\text{in}}\cdot\frac{S_x}{r(\boldsymbol{p}_i^c)\cdot S(v(\boldsymbol{p}_i^c))}, \tag{23}$$

where σ_{in} is an input parameter and S_x indicates the area calculated with the point $\boldsymbol{p}_x = [1, 0, 0]$, which just scales the resulting standard deviation σ_i to σ_{in} with the point $\boldsymbol{p}_x$.

By adjusting the standard deviation according to (23), the non-proportional displacement conversion problem can be handled as in Figure 9. In the figure, the Gaussian distribution with the adjusted standard deviation shows a revised influential range, which reflects the distance to the point and the perspective distortion of the image.

[5]The Gaussian center, which is the projection of the edge point, is a real number, whereas the surface area (21) is derived based on pixels represented by integers. Thus, we assume a virtual pixel at the projection of the edge point so that the area of surface can be calculated with non-integer values without loss of generality.

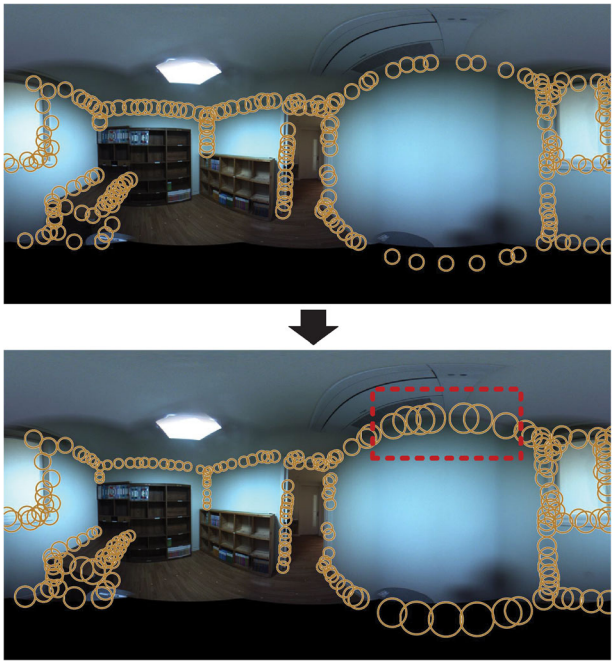

FIGURE 9 Projected edge points with uniform standard deviation (top) and adjusted standard deviation (bottom). Each circle indicates a radius of 3σ to a Gaussian center. In the top image, we set the uniform standard deviation to four, and σ_{in} is set to six in the bottom image. The adjusted standard deviation increases in the large-distortion region (red box) and decreases as the distance to the camera increases. With the adjusted standard deviation, each Gaussian distribution covers similar geometry and relaxes the non-proportional displacement conversion. It should be noted that we manually selected edge points on the walls and the desk for visualization purposes [Color figure can be viewed at wileyonlinelibrary.com]

6 | OPTIMIZATION

The goal of our calibration algorithm is to estimate the precise calibration parameters $\tau = [\boldsymbol{\omega},\ \boldsymbol{t}] \in \mathbb{R}^{6\times1}$. To obtain the optimal calibration parameters, we need to minimize the proposed cost function in Section 5 by iterative optimization methods, such as the gradient descent method.

The gradient descent method iteratively updates the calibration parameters in the negative gradient direction, and the gradient is calculated with respect to the current calibration parameters as follows:

$$\tau_{n+1} = \tau_n - \alpha\cdot\boldsymbol{g}(\tau_n), \tag{24}$$

where $\boldsymbol{g}(\tau_n) \in \mathbb{R}^{6\times1}$ indicates the gradient of the proposed cost at τ_n. Here, we calculate a step size α by the backtracking line search

algorithm to satisfy the Wolfe conditions (Wolfe, 1969), such that the resulting step size decreases the cost and the gradient. The Wolfe conditions are:

$$\mathrm{GMM}(\tau_{n+1}) \leq \mathrm{GMM}(\tau_n) - c_1 \cdot \alpha \boldsymbol{g}^T(\tau_n)\boldsymbol{g}(\tau_n), \tag{25}$$

$$\boldsymbol{g}^T(\tau_n)\boldsymbol{g}(\tau_{n+1}) \leq c_2 \cdot \boldsymbol{g}^T(\tau_n)\boldsymbol{g}(\tau_n), \tag{26}$$

where $c_1 \in (0, c_2)$ is adjusted to become very small compared to $c_2 \in (c_2, 1)$ (Nocedal & Wright, 1999). In this study, we set c_1 and c_2 to 10^{-4} and 0.9, respectively.

Moreover, the gradient $\boldsymbol{g}(\tau)$ of the proposed GMM cost in (24), (25), and (26) with respect to the set of calibration parameters τ can be represented as

$$\frac{\partial \mathrm{GMM}(\tau)}{\partial \tau} = -\sum_{i=1}^{N_P} \sum_{j \in \Omega_i} w_{i,j} \cdot \frac{\partial \boldsymbol{G}(d_{i,j}, \sigma_i)}{\partial \tau}, \tag{27}$$

which is the weighted sum of the Gaussian gradients with respect to τ. Here, we can calculate the Gaussian gradient by a combination of the derivatives with respect to the Gaussian center $[u_i,\ v_i]$ as follows:

$$\frac{\partial \boldsymbol{G}(d_{i,j}, \sigma_i)}{\partial \tau} = \frac{\partial \boldsymbol{G}(d_{i,j}, \sigma_i)}{\partial u_i} \cdot \frac{\partial u_i}{\partial \tau} + \frac{\partial \boldsymbol{G}(d_{i,j}, \sigma_i)}{\partial v_i} \cdot \frac{\partial v_i}{\partial \tau}. \tag{28}$$

In this equation, we obtain the Gaussian gradients with respect to each Gaussian center $\partial \boldsymbol{G}(d_{i,j}, \sigma_i)/\partial u_i$ and $\partial \boldsymbol{G}(d_{i,j}, \sigma_i)/\partial v_i$ by differentiating the Gaussian kernel (9) with $d_{i,j}$ (the distance from the Gaussian center $[u_i,\ v_i]$ to each edge pixel).

Moreover, the u-coordinate of the Gaussian center u_i is a function of $\boldsymbol{p}_i^c$, as in (15). Thus, the gradient of u_i with respect to τ is

$$\frac{\partial u_i}{\partial \tau} = \frac{\partial u_i}{\partial x_i^c} \cdot \frac{\partial x_i^c}{\partial \tau} + \frac{\partial u_i}{\partial y_i^c} \cdot \frac{\partial y_i^c}{\partial \tau} + \frac{\partial u_i}{\partial z_i^c} \cdot \frac{\partial z_i^c}{\partial \tau}, \tag{29}$$

where the derivative of u_i with respect to each coordinate of $\boldsymbol{p}_i^c$ can be calculated from (15). In the same way, the gradient of v_i can also be derived using (16).

Then, we need to calculate the rest of (29), which are gradients of each coordinate of $\boldsymbol{p}_i^c$ with respect to τ. Among them, we can easily determine the derivative with respect to the translation from (3). Moreover, the derivative with respect to the rotation vector $\boldsymbol{\omega} = [\omega_1,\ \omega_2,\ \omega_3] \in \mathfrak{so}(3)$ can be derived according to Barfoot (2017) as follows:

$$\begin{aligned} \frac{\partial \boldsymbol{p}_i^c}{\partial \boldsymbol{\omega}} &= \frac{\partial (\boldsymbol{R} \cdot \boldsymbol{p}_i + \boldsymbol{t})}{\partial \boldsymbol{\omega}} \\ &= -(\boldsymbol{R} \cdot \boldsymbol{p}_i)^{\wedge} \boldsymbol{J}, \end{aligned} \tag{30}$$

where the i-th column indicates the derivative with respect to the rotation parameter ω_i and $\boldsymbol{J}$ represents the left Jacobian of $SO(3)$:

$$\boldsymbol{J} = \frac{\sin(\bar{\omega})}{\bar{\omega}} \boldsymbol{I} + \left(1 - \frac{\sin(\bar{\omega})}{\bar{\omega}}\right) \boldsymbol{a}\boldsymbol{a}^T + \frac{1 - \cos(\bar{\omega})}{\bar{\omega}} \boldsymbol{a}^{\wedge}, \tag{31}$$

where $\bar{\omega} = |\boldsymbol{\omega}|$ and $\boldsymbol{a} = \boldsymbol{\omega}/\bar{\omega}$.

Finally, we can derive the gradient of the proposed cost in (27) by combining the derivatives in (28)–(31) to optimize the proposed cost function analytically by the gradient descent method.

Then, with the derived gradient, we perform an optimization in a coarse-to-fine refinement scheme by gradually decreasing the parameter σ_{in} to enhance the robustness of the proposed method. In particular, the coarse estimation of the calibration parameters is conducted first by optimizing the $GMM(\tau)$ with large σ_{in}. Here, the large standard deviation smooths the cost function such that the calibration parameters can be robustly modified from the initial guess. Then, we utilize the optimized calibration parameters as the initial guess of a subsequent optimization and refine it with the decreased σ_{in}. The proposed algorithm repeats this procedure according to a predefined maximum number of refinements (three in our case).

We also employ multiple pairs of images and point clouds to increase the robustness of the optimization result. The accumulation of measurements allows us to obtain edge pixels and edge points at various positions, similarly to taking checkerboard images from diverse points of view to calibrate camera-intrinsic parameters, as in Bouguet (2015). With the edges acquired at various positions, the proposed algorithm can avoid convergence to inaccurate local minima by preventing undesired outliers from dominating the calibration results.

7 | METHOD

In this section, we describe the implementation details of the proposed calibration method, as shown in Algorithm 1.

The inputs of Algorithm 1 are the measurements I and P of each sensor, which are obtained at n different locations; the initial guess τ_0 of the calibration parameters; and the parameter σ_{all}, which adjusts the standard deviation of each refinement level, whereas the output is the optimized calibration parameters. In the first step, we set the calibration parameters to our initial guess.

In the second step, we extract the edge score of each measurement for the preprocessing. Here, an edge score of the image is computed as discussed in Section 4.1, and an edge score of the point cloud is calculated according to (7).

In the third step, the algorithm optimizes the proposed cost function in a coarse-to-fine scheme by gradually decreasing σ_{in}. At each level, the gradient descent method minimizes the proposed cost function with the corresponding σ_{in}, and the function GetCost$(\cdot)$ calculates the proposed cost as in (8). Furthermore, the gradient of the cost with respect to the current calibration parameters is acquired by the function GetGradient$(\cdot)$, as discussed in (27)–(31). Then, the function GetStepSize$(\cdot)$ determines the amount of an update in the gradient direction to satisfy the Wolfe conditions (25) and (26).

The procedure detailing the cost function calculation is presented in Algorithm 2. Here, the cost is calculated for each pair of pixels and points if both the pixel and the point indicate

Algorithm 1 Gaussian Mixture Model-Based Calibration

Notations
n: The number of the input pairs.
k: The maximum number of coarse-to-fine refinement.

Inputs
$I :=$ Set of rectified grayscale images, $\{I_1, I_2, ..., I_n\}$.
$P :=$ Set of point clouds, $\{P_1, P_2, ..., P_n\}$.
$\boldsymbol{\tau}_0 \in \mathbb{R}^{6\times 1} :=$ Initial guess of calibration parameters.
$\sigma_{all} \in \mathbb{R}^{k\times 1} :=$ Set of σ_{in} in (23) of each refinement level, which decreases as refinement level increases.

Outputs
$\boldsymbol{\tau} \in \mathbb{R}^{6\times 1}$: Estimated calibration parameters.

Step1) Initialize
$\boldsymbol{\tau} = \boldsymbol{\tau}_0$;

Step2) Edge score extraction
$E_I = \text{GetImageEdgeScore}(I)$;
$E_P = \text{GetPointEdgeScore}(P)$;

Step3) Optimization
for level $= 1 : k$ **do**
  $\sigma_{in} = \sigma_{all}$ (level);
  while not converge($\boldsymbol{\tau}$) **do**
    $c = \text{GetCost}(I, P, E_I, E_P, \boldsymbol{\tau}, \sigma_{in})$;
    $\boldsymbol{g} = \text{GetGradient}(I, P, E_I, E_P, \boldsymbol{\tau}, \sigma_{in})$;
    $\alpha = \text{GetStepSize}(c, I, P, E_I, E_P, \boldsymbol{\tau}, \sigma_{in})$;
    $\boldsymbol{\tau} = \boldsymbol{\tau} - \alpha \cdot \boldsymbol{g}$;
  end while
end for

edge scores higher than the predefined thresholds. To calculate the cost, the function GetWeight(·) obtains the weight w, as in (10). Additionally, the function Projection(·) projects the point $\boldsymbol{p}$ onto the image, as in (15) and (16), to acquire $\boldsymbol{p}_{\text{proj}}$. Then, the displacement d of the projected point $\boldsymbol{p}_{\text{proj}}$ and the pixel $\boldsymbol{q}$ is calculated as the shortest distance between them on the sphere by the function GetDistance(·), as discussed in Section 5.2. Next, the function GetSigma(·) calculates the standard deviation of the Gaussian distribution, as in (23). Finally, if the displacement is within the three-sigma of the Gaussian distribution, the cost of the current point $\boldsymbol{p}$ and the current pixel $\boldsymbol{q}$ is calculated as the negative of the weighted Gaussian distribution, as proposed in (8).

8 | EXPERIMENTAL RESULTS

In this section, we validate the performance of the proposed calibration algorithm experimentally and compare it to other calibration methods.

8.1 | Dataset

To test the algorithm performance in various environments, we employed three datasets[6] with different characteristics as follows:

8.1.1 | Indoor dataset

The dataset consists of 63 pairs of images and point clouds of different locations in a static indoor environment, as in Figure 10a. As figure shows, the indoor dataset includes many edges from the walls, floor, and ceiling.

To acquire the 3D point clouds, we used a Velodyne VLP-16 LIDAR, which scans the environment by horizontally rotating 16 laser beams with a vertical field of view between −15° and 15° and a resolution of 2°. The image was taken by the Point Grey Ladybug5 camera, which consists of six CCD sensors to acquire images in various directions. In this study, we used a spherical panoramic image of size 1,024 × 512 for the calibration.

[6]Our datasets (indoor and outdoor) are publicly available at https://doi.org/10.5281/zenodo.2640062

Algorithm 2 Cost Function

```
function GetCost(I, P, E_I, E_P, τ, σ_in)
    cost = 0;
    for each pair of edge pixels and edge points do
        q = GetCurrentPixel(I);
        p = GetCurrentPoint(P);
        if E_I(q) > threshold_I and E_P(p) > threshold_P then
            w = GetWeight(q, p, E_I, E_P);
            p_proj = Projection(p, τ);
            d = GetDistance(p_proj, q);
            σ = GetSigma(p, τ, σ_in);
            if d ≤ 3σ then
                cost = cost − w·GaussKernel(d, σ);
            end if
        end if
    end for
    return cost;
end function
```

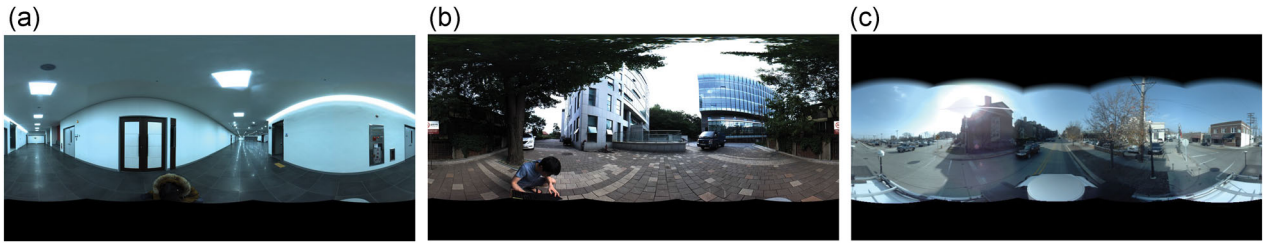

FIGURE 10 Environmental characteristics of the datasets used for the performance evaluation. The figures represent the acquired images of the (a) indoor, (b) outdoor, and (c) Ford datasets. The indoor environment consists of more edges from indoor structures compared to the outdoor and Ford datasets. Regarding the environment scale, the indoor dataset is the smallest, because measurements were collected in a limited indoor area. Additionally, the outdoor dataset was obtained in a smaller environment compared to the Ford dataset. In particular, unlike the other datasets, the Ford dataset was acquired in a less controlled condition (continuous motion and fully dynamic environment) [Color figure can be viewed at wileyonlinelibrary.com]

During the data acquisition, we rigidly mounted the VLP-16 and Ladybug5 on the sensor frame, as in Figure 2. Additionally, we acquired each pair of images and point clouds with the sensor system standing still, preventing deviations due to inaccurate synchronization.

8.1.2 | Outdoor dataset

To test the algorithm performance in a less structured environment, we collected 61 pairs of images and point clouds in an outdoor environment, as shown in Figure 10b. The outdoor environment had fewer edge features in a larger space compared to the indoor environment. All the settings for the data acquisition were the same as those used for obtaining the indoor dataset.

8.1.3 | Ford dataset

Pandey, McBride, and Eustice (2011) provide a public dataset, which contains point clouds and their corresponding pinhole images acquired in an urban environment, as shown in Figure 10c. Unlike our datasets, the sensor system collects measurements while it is continuously moving in a fully dynamic environment.

The point clouds of the Ford dataset were acquired by the Velodyne HDL-64E LIDAR, which has 64 rays with 26.9° vertical field of view (providing denser measurements compared to the VLP-16 of our sensor system). The point clouds are then motion-compensated by the high-end IMU (POS-LV 420) to relax the distortion due to continuous motion. Additionally, pinhole images are obtained by the five horizontal cameras of the Ladybug3. We generated a spherical panoramic image (1,024 × 512) from the provided pinhole images by using the manufacturer's intrinsic parameters and the Ladybug software development kit.

8.2 | Performance comparison

In this section, we compare our method to current state-of-the-art techniques to evaluate the performance of the proposed

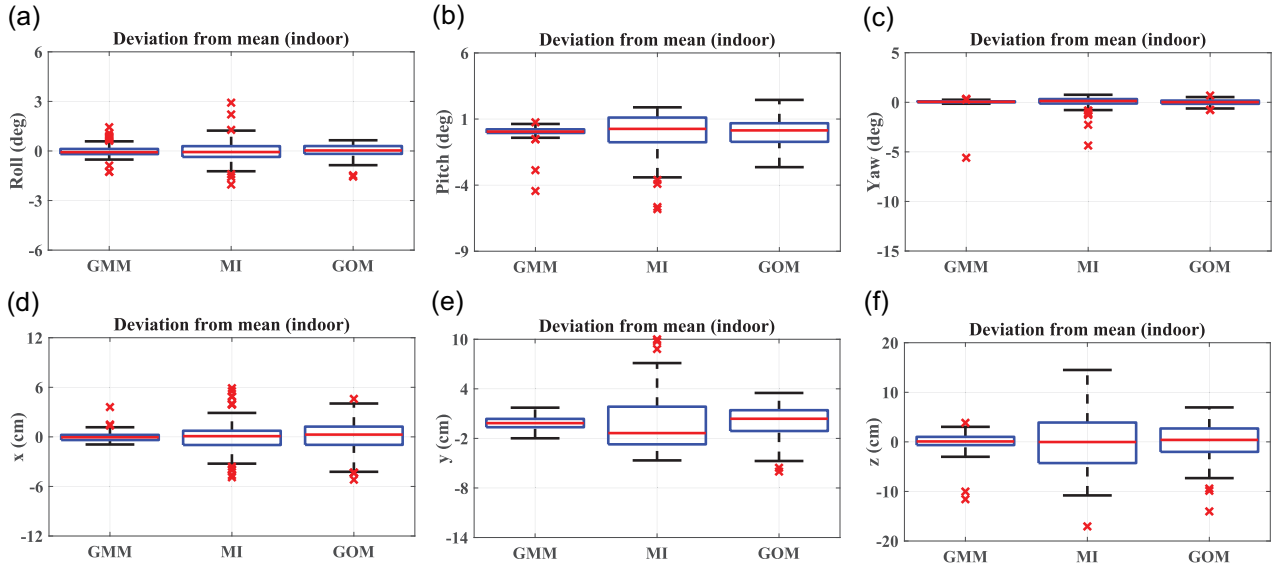

FIGURE 11 Deviation of calibration results with the random indexed input pairs of the indoor dataset. Each figure presents the deviation from the mean of the calibration parameters (a) roll, (b) pitch, (c) yaw, (d) t_x, (e) t_y, and (f) t_z that are estimated by the proposed GMM, MI, and GOM methods, as in Section 8.2.1. The boxes contain the calibration results between the 25th (Q_1) and 75th percentiles (Q_3), and the red lines in the boxes indicate the median values. Additionally, the whiskers end at the most distinct results within $1.5 \times (Q_3 - Q_1)$ from the boxes (Q_1 and Q_3). The red "×" symbol shows the results outside of the whiskers. The calibration results from the proposed method have lower deviation for all parameters [Color figure can be viewed at wileyonlinelibrary.com]

method. Specifically, we examine the performance of the proposed calibration algorithm regarding (i) *precision* and (ii) *accuracy*.

8.2.1 | Implemented methods

We implemented the proposed method and two previous methods for the performance comparison as follows.

- **GMM**: The method proposed in this study uses the GMM as in Algorithm 1 with σ_{in} of each coarse-to-fine refinement level equals to 15, 10, and 5 for the indoor and outdoor datasets and 25, 20, and 15 for the Ford dataset. The thresholds for image and point cloud edge scores are set to 0.15 and 0.10, respectively.
- **MI**: The MI-based calibration method proposed by Pandey et al. (2015). We conducted the calibration with the open source code provided by the authors. As in the original manuscript, we employed images from five Ladybug3/Ladybug5 horizontal pinhole cameras for the calibration.
- **GOM**: The calibration by maximizing the gradient orientation measure (GOM) metric proposed by Taylor et al. (2015). For the calculation of the metric, we used the return intensity as a feature. Additionally, we slightly modified the open source code implemented by the authors and applied the spherical camera model. We set the standard deviations of the Gaussian blur in the GOM algorithm to 12, 6, 3, and 0, which were empirically set to yield accurate results.

8.2.2 | Precision comparison

Precision measures the consistency of the calibration results for various inputs. This can be measured by the standard deviation of the optimized calibration parameters.

For this purpose, we randomly sampled 12 pairs of images and point clouds from the raw dataset (the indoor/outdoor/Ford dataset in Section 8.1) to create a set of input pairs. Each calibration method then optimized the calibration parameters by using the input set with the same initial guess for the parameters. This process was repeated 100 times with different randomly sampled input pairs from each dataset to obtain the calibration results with various inputs.

Figure 11 gives the indoor calibration results for each method obtained from the abovementioned procedure. The figure shows that the proposed method (GMM) converges to the more consistent values with various input measurements. Table 1 also shows the standard deviation for the indoor calibration results. As in the table, the proposed method results in lower standard deviations for all the calibration parameters in comparison to other methods, except for the yaw angle. However, as shown in Figure 11, the proposed GMM method shows lower deviation in the yaw angle without outliers. In other words, the results of the indoor dataset indicate that the calibration results of the proposed method are more consistent than those of other methods.

The outdoor calibration results are given in Figure 12. As presented in the figure, the MI method shows increased standard deviation compared to other methods. The standard deviations of pitch and yaw are the smallest when using the proposed method, as shown in Table 1, whereas the GOM method has the smallest standard deviation in roll. In translation, the GOM method has the

TABLE 1 Standard deviation with random sampled input pairs

Methods	Roll (deg)	Pitch (deg)	Yaw (deg)	t_x (m)	t_y (m)	t_z (m)
Indoor dataset						
GMM	0.386	0.586	0.572	0.006	0.014	0.021
MI	0.706	1.557	0.629	0.020	0.039	0.057
GOM	0.404	1.014	0.265	0.020	0.019	0.039
Outdoor dataset						
GMM	0.496	0.494	0.663	0.018	0.019	0.027
MI	2.262	5.978	4.536	0.047	0.034	0.042
GOM	0.179	1.207	4.046	0.014	0.016	0.021
Ford dataset						
GMM	0.303	0.568	0.223	0.013	0.011	0.067
MI	5.294	3.028	0.768	0.062	0.053	0.116
GOM	0.494	0.710	2.630	0.050	0.011	0.129

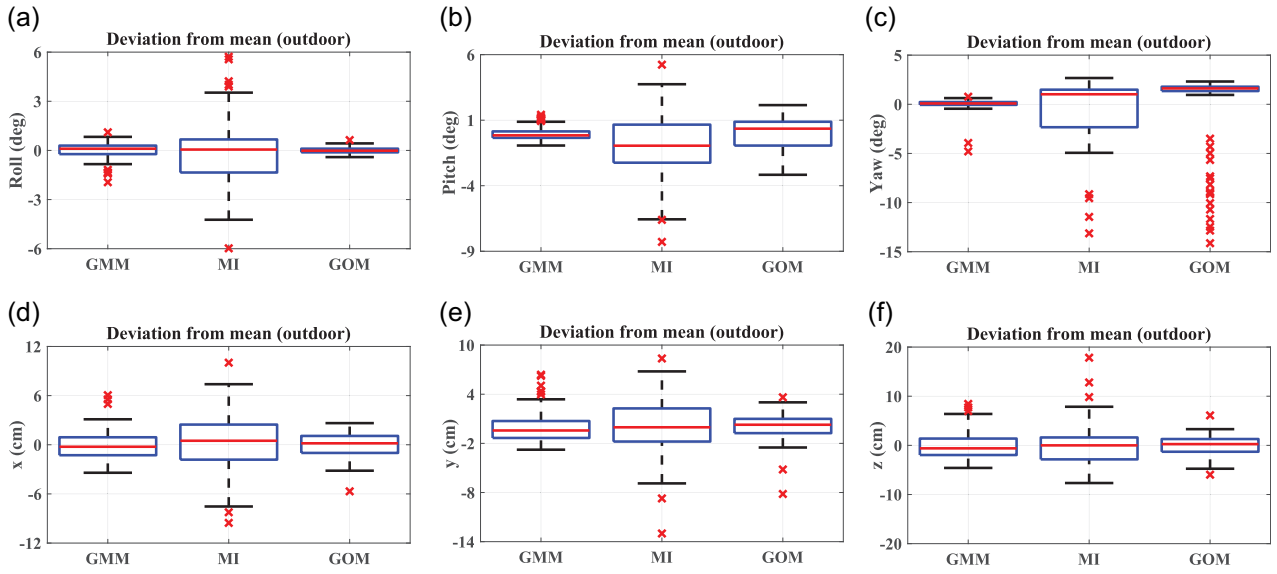

FIGURE 12 Deviation of calibration results with the random indexed input pairs of the outdoor dataset. The deviation from the mean of the calibration parameters (a) roll, (b) pitch, (c) yaw, (d) t_x, (e) t_y, and (f) t_z is depicted using the box plots as in Figure 11. The proposed method and the GOM method result in comparable deviations using the outdoor dataset [Color figure can be viewed at wileyonlinelibrary.com]

smallest standard deviations, but the standard deviations of the proposed method are also comparable. In other words, the outdoor results indicate that the proposed method and the GOM method have similar performance in precision.

Finally, the calibration results of the Ford dataset are shown in Figure 13. In the figure, the proposed method converges to more consistent calibration parameters compared to both the MI and GOM methods. Consequently, both methods show larger standard deviations compared to the proposed method, as shown in Table 1. From the results of the three different datasets, the proposed method shows consistent convergence, but it does not guarantee that the converged parameters are the correct solution. Therefore, we need to analyze the accuracy together to make a complete comparison of the algorithm performance.

8.2.3 | Accuracy comparison

As there is no available ground truth, the accuracy of the calibration result can be measured indirectly by calculating the distance between the projection of a point and a corresponding pixel in a 2D image. Ideally, this reprojection error can be acquired by manually selecting the distinct points from the point cloud and the corresponding pixels in the image. However, owing to the sparsity of the point cloud, it is difficult to find an exact one-to-one correspondence between a point and a pixel.

Thus, we manually found a line-to-line correspondence, which consists of points ($\boldsymbol{p}_1$, $\boldsymbol{p}_2$, ...) on a 3D line and pixels ($\boldsymbol{q}_1$, $\boldsymbol{q}_2$, ...) on the corresponding 2D line. Then, the reprojection error of point $\boldsymbol{p}_i$ is calculated as the shortest distance between the projection of the point

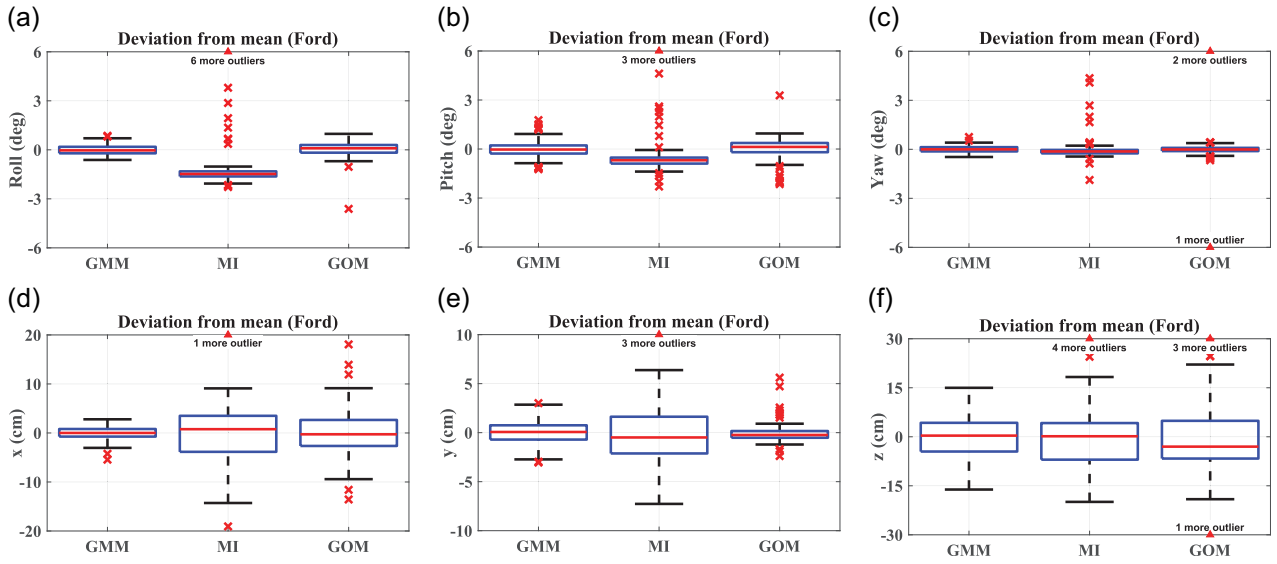

FIGURE 13 Deviation of calibration results with the random indexed input pairs of the Ford dataset. The deviation from the mean of the calibration parameters (a) roll, (b) pitch, (c) yaw, (d) t_x, (e) t_y, and (f) t_z is depicted using the box plot as in Figure 11. The red triangles in the figure represent the outliers that are off the axis of each figure. The proposed method provides more consistent results compared to the other methods [Color figure can be viewed at wileyonlinelibrary.com]

$\boldsymbol{q}(\boldsymbol{p}_i^c)$, as in (14) and all the pixels ($\boldsymbol{q}_1$, $\boldsymbol{q}_2$, ...) of the corresponding lines. This rule is similar to that of the iterative closest point algorithm (Besl & McKay, 1992), which defines the correspondence of a point from one point cloud as the closest point in the other point cloud to measure the displacement between the two point clouds.

For the calculation of the reprojection error, images and point clouds were collected independently from the dataset used in the calibration. The dataset used in the accuracy test consists of 15 pairs[7] of images and point clouds from various locations. Additionally, we increased the image resolution to 2,048 × 1,024 to improve the manual picking process.

From the accuracy test dataset, we manually picked 82, 94, and 100 line correspondences for the indoor, outdoor, and Ford results, respectively; we then calculated the previously described reprojection errors with the selected correspondences. Here, we projected the point by using the most representative calibration parameter set for each method to test the accuracy independently of the precision analysis.

The representative calibration parameter set is picked from the 100 parameter sets used for the precision analysis in Section 8.2.2. Specifically, a parameter set that had the minimum Mahalanobis distance using the covariance of the 100 parameter sets (i.e., the distance between a parameter set and a parameter distribution) is selected as the representative calibration parameter set for each method using each dataset. Table 2 shows the representative calibration parameters for each method according to the Mahalanobis distance. The table also contains the mean of the reprojection error calculated using each representative calibration parameter set.

Regarding the indoor and outdoor results in Table 2, the GMM method has lower reprojection errors compared to other methods. Furthermore, Figure 14 shows the histogram of the overall reprojection error and the reprojection errors for different scenes. The figures show that the GMM method reduces the overall reprojection error and produces more consistent reprojection errors in all the scenes of the indoor and outdoor environments. These results indicate that the proposed GMM method accurately and globally calibrates the transformation between the two sensors in both environments.

The point cloud textured by each representative calibration parameter set of the indoor results is also given in Figure 15 to visually verify the accuracy analysis. The point cloud of the proposed method in the figure shows that it is correctly aligned to the color measurements compared to the other methods, indicating that the proposed method results in accurate parameter estimation.

When using the Ford dataset, the proposed GMM method shows 2.8% and 8% reduced reprojection error overall compared to the MI and GOM methods, as shown in Table 2. Therefore, considering the precision and accuracy results of Tables 1 and 2 together, it is clear that the proposed method also provides enhanced performance using the Ford dataset.

As presented in Table 2 and Figure 14, the outdoor results show increased reprojection error compared to the indoor results. This is because of the dynamic and featureless characteristics of the outdoor environment, which interfere with accurate calibration. It also should be noted that the accuracy enhancement of the proposed method using the Ford dataset is less than those of the indoor and

[7]When collecting the raw datasets in the indoor and outdoor environments, we reassembled the sensor system each time. Thus, two different sets of 15 pairs were obtained to analyze the accuracy of the indoor and outdoor results.

TABLE 2 Reprojection error with calibration results

Methods	Roll (deg)	Pitch (deg)	Yaw (deg)	t_x (m)	t_y (m)	t_z (m)	Reprojection error (pixel)
Indoor dataset							
GMM	−0.18	15.47	−55.14	0.0934	0.0597	−0.1659	**1.78**
MI	0.12	14.85	−55.20	0.0840	0.0938	−0.2033	**3.33**
GOM	1.75	14.57	−55.13	0.1359	0.1411	−0.2377	**7.23**
Outdoor dataset							
GMM	−0.39	16.54	−55.35	0.0950	0.0830	−0.1670	**3.83**
MI	2.02	13.52	−55.33	0.0979	0.1798	−0.2197	**10.42**
GOM	2.07	14.76	−55.03	0.1351	0.1429	−0.2193	**8.45**
Ford dataset							
GMM	0.98	−0.78	−88.88	0.3034	0.0257	−0.2480	**7.85**
MI	−0.02	−0.27	−89.05	0.2695	−0.0044	−0.3059	**8.07**
GOM	1.87	1.57	−88.63	0.3477	0.0327	−0.5039	**8.53**

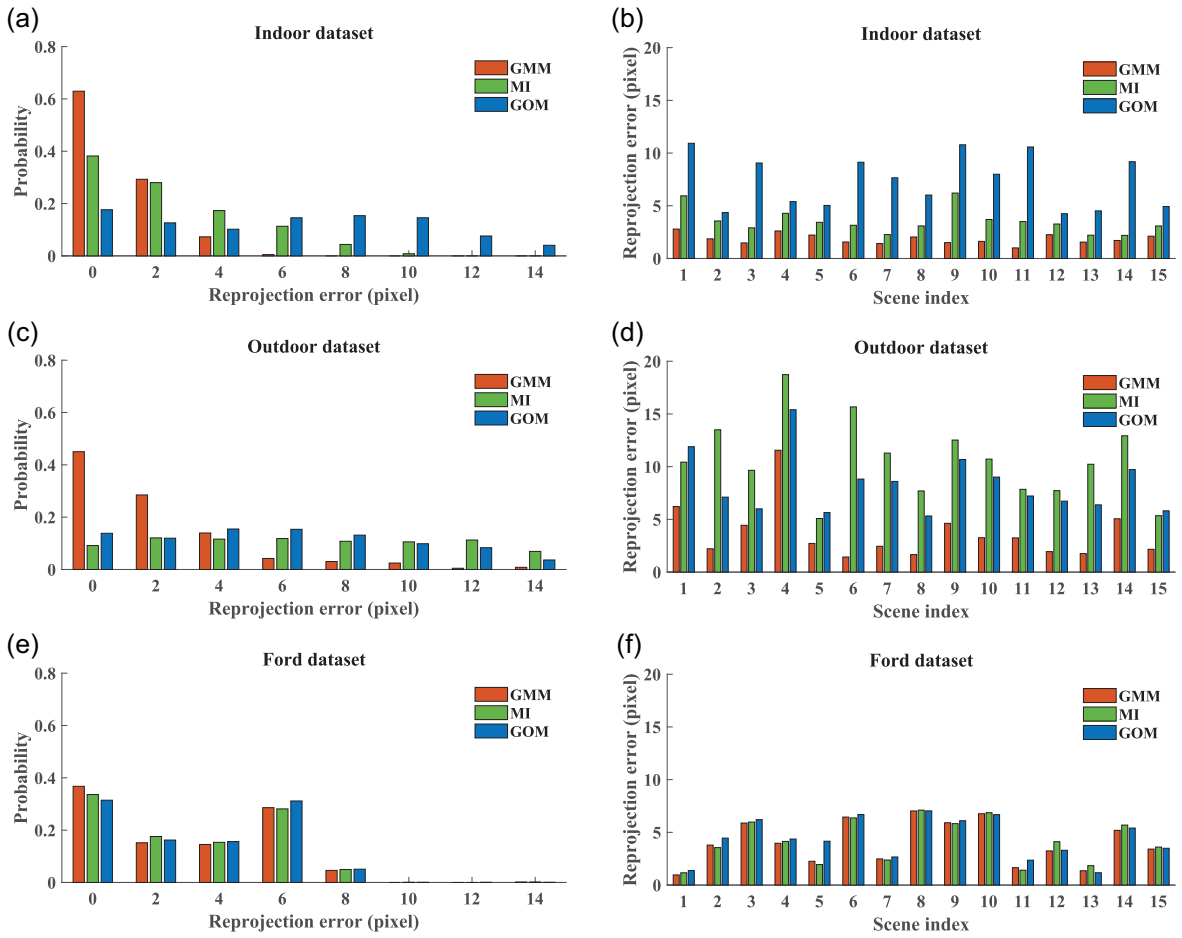

FIGURE 14 Reprojection errors of the indoor dataset (82 line correspondences), outdoor dataset (94 line correspondences), and Ford dataset (100 line correspondences). We calculated each reprojection error using the parameters from the GMM, MI, and GOM methods. In (a), (c), and (e), the histograms of the overall reprojection error are given, and (b), (d), and (f) show 15 means of reprojection errors per scene in each environment. The proposed GMM method results in decreased overall reprojection errors and consistent reprojection errors for every scene in the indoor and outdoor environments. When using the Ford dataset, the proposed GMM method shows reprojection errors comparable to the other methods [Color figure can be viewed at wileyonlinelibrary.com]

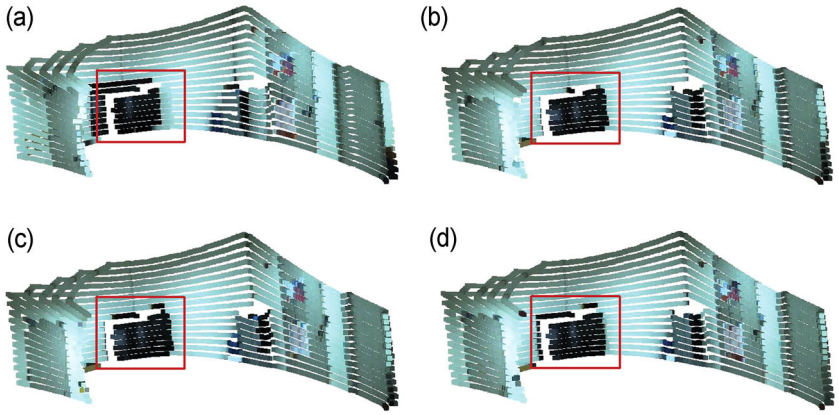

FIGURE 15 Textured point cloud using indoor calibration results for each method. Each figure represents the textured point cloud by the calibration parameters of the (a) initial estimate, (b) GMM, (c) MI, and (d) GOM. For (b)–(d), we textured the point cloud by employing the representative calibration parameters of each method as in Table 2. The proposed GMM method accurately enhances the boundary consistency of the environment in the red box [Color figure can be viewed at wileyonlinelibrary.com]

outdoor datasets. Two key reasons, induced by data acquisitions on a moving car, are asynchronous time stamps of measurements (images and point clouds) and motion distortion, which degrade calibration performance.

8.3 | Analysis of the proposed cost function

In Section 5, we proposed the GMM cost function (8), which accounts for the intensity of an edge by the weight in (10) and the influential range of an edge by the standard deviation control in (23). In this section, we verify the usefulness of the proposed (i) weight and (ii) standard deviation techniques for improving calibration performance.

To that end, we performed the calibration by minimizing the GMM cost, the non-weighted GMM cost, and the uniform standard deviation GMM cost. Here, we used 100 randomly sampled input pairs from the indoor dataset as in Section 8.2.2, which we call the random indexed input pairs in the following sections, and obtained 100 sets of calibration parameters using each method.

In the non-weighted GMM cost calculation, the weight term in (10) was fixed at one for all the edge pairs. Additionally, the standard deviation term in (23) was constant for all the Gaussian distributions in the calculation of the uniform standard deviation GMM cost. Then, we analyze the performance of each method as follows:

First, parameters of the GMM cost and the non-weighted GMM cost are presented in Figure 16. In the figure, the calibration parameters of the non-weighted GMM cost show large fluctuations. To filter the outliers, we used a criterion whereby calibration parameters deviating from the median of the GMM results (100 parameter sets) by more than 2° or 4 cm were labeled as outliers. This criterion labels 50% of the non-weighted calibration results as outliers, whereas the proposed GMM method results in 2% outliers. Therefore, we can conclude that the proposed weight calculation process remarkably enhances the quality of the optimized parameters.

Next, Figure 17 presents the results of the GMM cost and the uniform standard deviation GMM cost. In the latter case, we set the uniform standard deviation (σ_i) to 7.5, 5, and 2.5 for each coarse-to-fine level, which is empirically found to produce best calibration results. As shown in the figure, the uniform standard deviation GMM cost has more outliers (7% of the results) compared to the GMM cost (2%) under the same criterion as in the previous paragraph.

In particular, owing to the sparse angle resolution of the LIDAR (2°) and its limited field of view (30°) in the vertical direction, the z-directional translation is difficult to estimate with our sensor system, as the standard deviation of t_z shows in Table 1. Accordingly, the uniform standard deviation GMM cost produces substantial variation in the translation along the z-axis compared to the consistent results of the GMM cost, as shown in figure. This experimental result supports the usefulness of the proposed standard deviation control technique, which improves the robustness of the calibration results.

8.4 | Analysis of an analytical derivative

The proposed cost function is analytically differentiable, as we showed in Section 6. In general, the analytically calculated gradient leads to more accurate and reliable results compared to the numerically approximated gradient.

To demonstrate the advantage of employing the analytical gradient, we compared the calibration results obtained by the gradient descent method with analytical and numerical gradients. Here, we employed the optimization toolbox of MATLAB (The MathWorks Inc., 2017) for the numerical optimization where the numerical gradient is acquired using the forward finite difference approximation with the step size of 0.3° and 5 mm.

Table 3 presents a comparison of the calibration results for the random indexed input pairs, as in Section 8.3. As shown in the table, optimization with the analytical gradient gives better

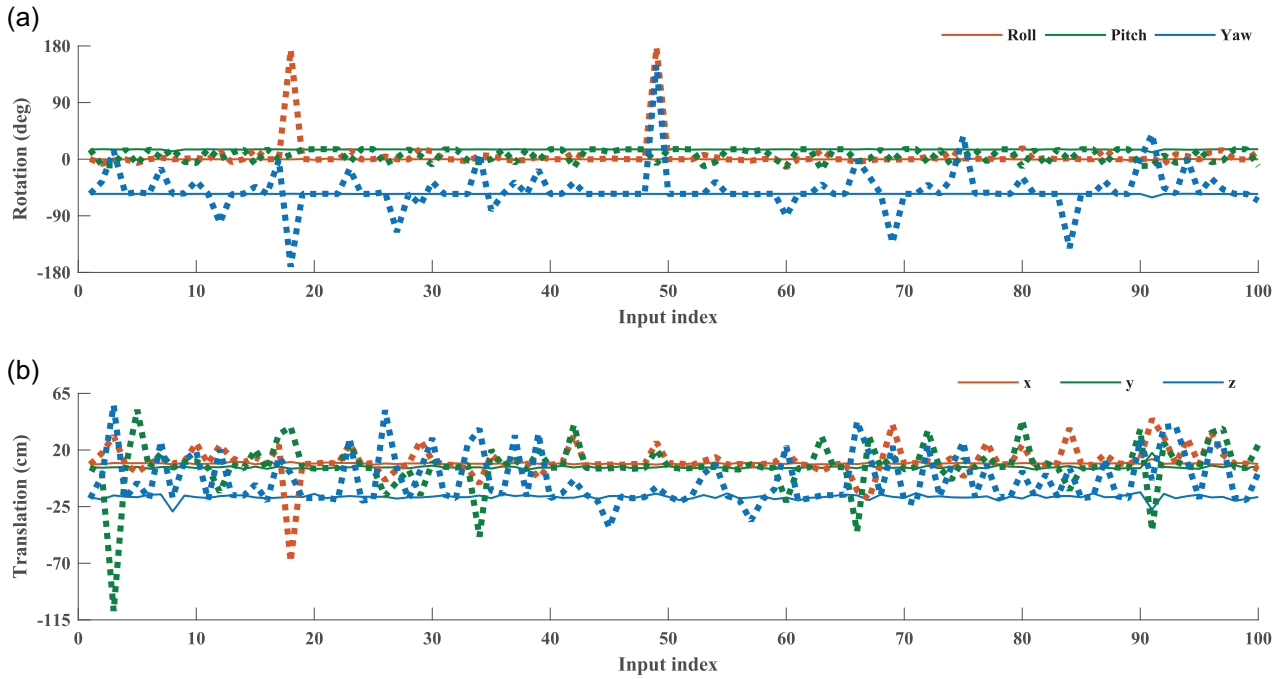

FIGURE 16 Calibration parameters using the proposed GMM cost and the non-weighted GMM cost. The calibration results of (a) rotation and (b) translation with the random indexed input pairs are shown as solid lines for the GMM cost and as dashed lines for the non-weighted GMM cost. The results of the non-weighted GMM have wider variation than the results of the GMM. From these figures, we can empirically validate the claim that the weight scheme (10) improves the robustness of the calibration results [Color figure can be viewed at wileyonlinelibrary.com]

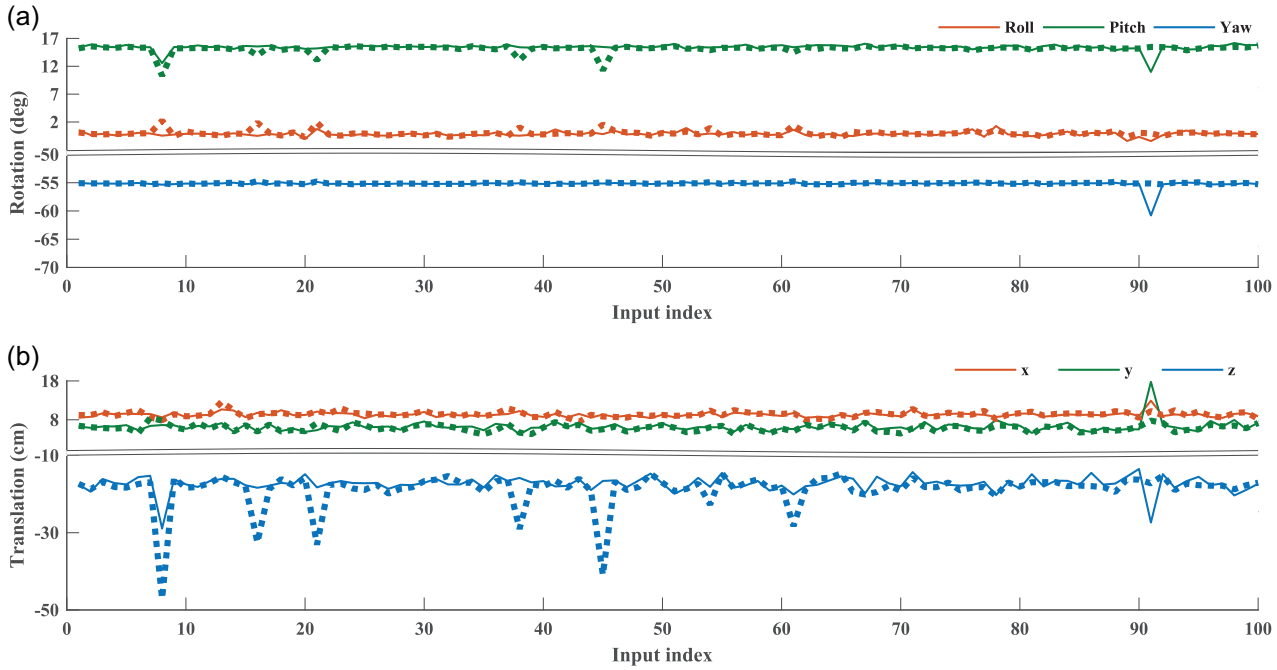

FIGURE 17 Calibration parameters from the proposed GMM cost and the uniform standard deviation GMM cost. As in the analysis of the non-weighted GMM cost, we employed random indexed input pairs. The calibration results of rotation and translation are given in (a) and (b), respectively. Here, the solid line shows the result of the GMM cost, and the dashed line represents the result of the uniform standard deviation GMM cost. As in the figures, the uniform standard deviation GMM cost shows less consistent results, especially in the z-directional translation. The results show that the application of the standard deviation control (23) leads to a meaningful improvement in the calibration results [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 3 Standard deviation and reprojection error with analytical and numerical gradients

Methods	Roll (deg)	Pitch (deg)	Yaw (deg)	t_x (m)	t_y (m)	t_z (m)	Reprojection error (pixel)
Analytical	0.386	0.586	0.572	0.006	0.014	0.021	1.78
Numerical	1.997	2.912	1.997	0.015	0.034	0.037	3.34

results in terms of both the standard deviation (precision) and the reprojection error (accuracy). In conclusion, we have experimentally validated the claim that the analytically differentiable property of the proposed cost function boosts the performance of the proposed algorithm.

8.5 | Convergence analysis

In this section, we analyze the convergence property of the proposed calibration algorithm by (i) examining the convergence basin of the proposed cost function and (ii) analyzing the calibration results with different initial guesses.

For a stable convergence with iterative optimization techniques, the cost function needs to be convex around the solution with an adequate convergence basin. In this study, we investigated the convergence basin of the cost function by altering the rotation and translation from the optimized point.

Figure 18 presents the calculated costs with the rotation varying from −20° to 20° and the translation varying from −20 to 20 cm. In the figure, the proposed GMM cost shows convexity around the solution with a sufficient basin of convergence. In other words, the results demonstrate that the proposed cost function has sufficient convexity for the optimization of the transformation parameters.

The determination of an initial guess also has a significant effect on the calibration result, in that the optimization variables are updated based on the initial guess owing to the nature of the iterative optimization method. However, if the algorithm is to be sufficiently robust, the optimized results should not be dominated by initial guesses. In this context, we examined the proposed algorithm for various initial guesses to show its robustness to the initial guess.

To that end, we generated 200 initial guesses by adding randomly generated uniform noise to the optimized parameters. We set the uniform noise to vary between −5° and 5° in rotation and between −10 and 10 cm in translation, which is a reasonable amount of variation that captures the errors of a roughly measured initial guess. Then, the calibration was conducted to refine the obtained initial guesses.

Figure 19 shows the calibration results with the acquired initial guesses. In the figure, 98.5% of trials converge to consistent calibration parameters with standard deviations of less than 0.07° and 0.68 cm, except for t_z, which shows an increased standard deviation due to LIDAR's sparse resolution in the vertical direction. In conclusion, the results indicate that the proposed algorithm has sufficient robustness to compensate for the errors in reasonable initial guesses.

9 | DISCUSSION

In this study, we found that the GMM framework is beneficial for calibrating a camera-LIDAR sensor system, especially when the components of the GMM (weight, displacement, and standard deviation) are reasonably derived to represent the measurement properties. We also showed that the continuous nature of the GMM is suitable for managing the many-to-many correspondences, which are adaptively gathered by considering the projection characteristics.

The basic assumption of the proposed method is that the edge features of both measurements are highly correlated when the calibration parameters are correct. Consequently, the performance of the proposed algorithm is dependent on the environmental characteristics. In an edge-abundant indoor or urban outdoor environment, the proposed method can satisfactorily calibrate a camera-LIDAR sensor system. However, occasionally, sensors are required to acquire measurements in an environment with insufficient edge features, such as a forest, for other purposes. In this case, we predict the performance degradation of the proposed algorithm. Therefore, the calibration needs to be independently completed in an edge-existing region before acquiring measurements in an edgeless environment.

Algorithm complexity is another aspect we want to discuss in this section. The proposed projection model-based many-to-many correspondence scheme is beneficial for algorithm performance, but has a drawback in terms of algorithm complexity. In particular, the algorithm complexity of the proposed cost is $\mathcal{O}(N_l \times N_p)$ with N_l edge pixels and N_p edge points (see Equation (8)). Thus, the GMM requires more computation compared to the previous methods with linear complexity, which originates from one-to-one correspondences (after blurring the images in the case of Levinson & Thrun, 2013; Taylor et al., 2015). However, it is a good trade-off between algorithm complexity and algorithm performance, as the sensor calibration mostly conducted offline.

10 | CONCLUSION

We proposed a targetless calibration algorithm based on the cost function that evaluates the alignment of the many-to-many correspondence between edge pixels and projected edge points to precisely estimate the rigid-body transformation between a camera and LIDAR.

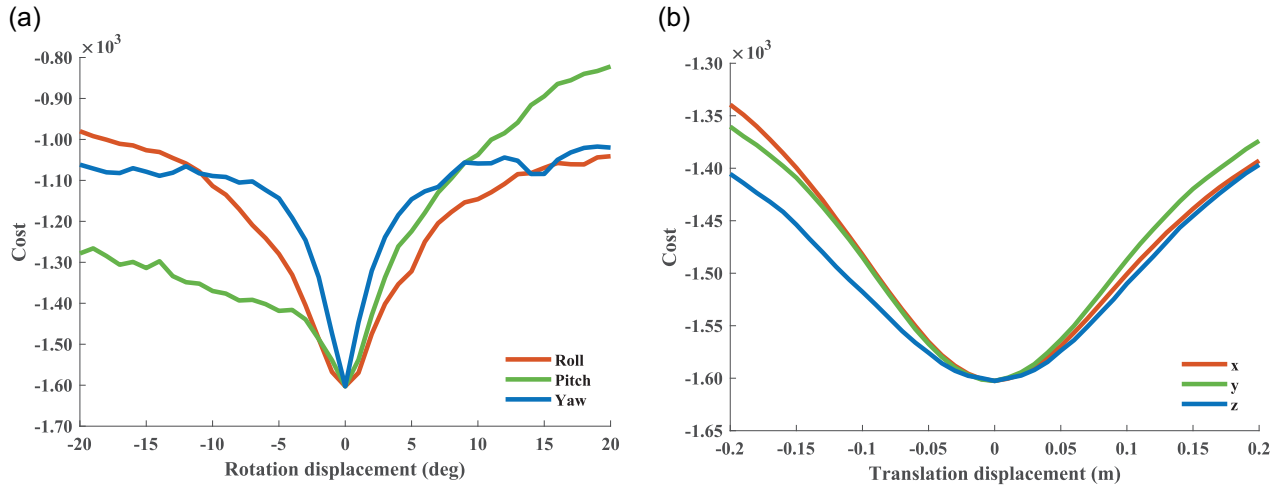

FIGURE 18 Convexity test of the proposed cost. Each figure presents the calculated costs by varying the (a) rotation and (b) translation from the optimized transformation, respectively. Here, we employed an input set of 12 pairs of images and point clouds. In both figures, the evaluated costs show a wide convex region around the solution [Color figure can be viewed at wileyonlinelibrary.com]

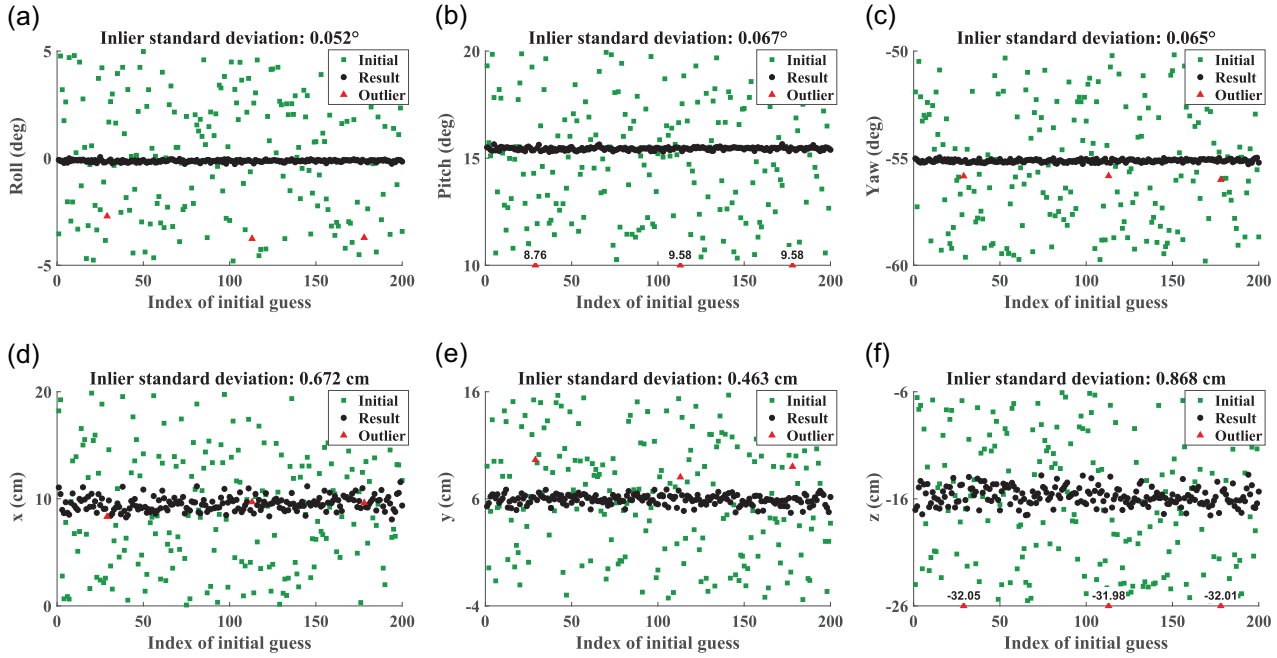

FIGURE 19 Calibration results with different initial guesses, where each figure presents (a) roll, (b) pitch, (c) yaw, (d) t_x, (e) t_y, and (f) t_z. The green squares indicate the 200 randomly obtained initial guesses, which vary from −5° to 5° in rotation and from −10 to 10 cm in translation. The black circles also show optimized results after calibration by using the corresponding initial guess. Additionally, the red triangles represent the results from the outlier parameter set. A parameter set (τ_i, $i \in [1, 200]$) is labeled an outlier if the distances between each parameter in the set and the median of the corresponding parameter exceed 0.5° or 2.5 cm. With these criteria, the three parameter sets (1.5%) of the overall results are filtered as outliers. In (b) and (f), the outliers that exceed the minimum of the initial guess are drawn on the x-axis with their values for visualization purposes. Except for the outliers, the proposed algorithm converges to consistent values with different initial guesses [Color figure can be viewed at wileyonlinelibrary.com]

To extract edge measurements, we introduced the edge score metric for the point cloud, which produces consistent results for both boundary-edge and fold-edge points, and we obtained edge pixels by applying the conventional gradient detection algorithm to the smoothed image.

Additionally, the proposed cost function is constructed based on the GMM framework to align the projection model-based many-to-many correspondences of edge pixels and edge points for an accurate and robust calibration. The proposed cost is also constructed to capture the intensity, location, and influential range of each measurement by the

weight, displacement, and standard deviation of the GMM, respectively. Moreover, the GMM framework provides the analytical derivative of the cost, which improves the accuracy of the optimization.

The experimental results show that the proposed calibration method outperforms other methods in various environments. Our analysis also reveals that the introduced weight, standard deviation techniques, and analytical derivative improve the performance of the proposed algorithm.

Given these results, we believe that the proposed calibration algorithm provides accurate calibration parameters that can assist various algorithms in utilizing camera and LIDAR systems.

ACKNOWLEDGMENTS

This study was partly supported by a National Research Foundation of Korea (NRF) Grant (No. 2011-0031648) funded by the Ministry of Science and ICT (MSIT), a Korea Evaluation Institute of Industrial Technology (KEIT) Grant (No. 10073166) funded by the Ministry of Trade, Industry and Energy (MOTIE), and a Korea Agency for Infrastructure Technology Advancement (KAIA) Grant (No. 18NSIP-B135768-02) funded by the Ministry of Land, Infrastructure and Transport (MOLIT), Republic of Korea.

ORCID

Jaehyeon Kang http://orcid.org/0000-0001-9913-644X
Nakju L. Doh http://orcid.org/0000-0002-7821-9809

REFERENCES

Agamennoni, G., Fontana, S., Siegwart, R. Y., & Sorrenti, D. G. (2016). Point clouds registration with probabilistic data association. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Daejeon, Korea (pp. 4092–4098).

Barfoot, T. D. (2017). *State estimation for robotics*. Cambridge, England: Cambridge University Press.

Besl, P. J., & McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(2), 239–256.

Bouguet, J.-Y. (2015). *Camera calibration toolbox for matlab*. Retrieved from http://www.vision.caltech.edu/bouguetj/calib_doc/.

Douillard, B., Fox, D., Ramos, F., & Durrant-Whyte, H. (2011). Classification and semantic mapping of urban environments. *International Journal of Robotics Research*, *30*(1), 5–32.

Geiger, A., Moosmann, F., Car, Ö., & Schuster, B. (2012). Automatic camera and range sensor calibration using a single shot. In *IEEE International Conference on Robotics and Automation*, Saint Paul, Minnesota (pp. 3936–3943).

González, A., Vázquez, D., López, A. M., & Amores, J. (2017). On-board object detection: Multicue, multimodal, and multiview random forest of local experts. *IEEE Transactions on Cybernetics*, *47*(11), 3980–3990.

Harris, C., & Stephens, M. (1988). A combined corner and edge detector. In *Alvey Vision Conference*, Manchester, UK (pp. 147–151).

Harris, J. W., & Stöcker, H. (1998). *Handbook of mathematics and computational science* (pp. 107–108). New York, NY: Springer.

Levinson, J., & Thrun, S. (2013). Automatic online calibration of cameras and lasers. In *Robotics: Science and Systems*, Berlin, Germany (pp. 29–36).

Manton, J. H. (2002). Optimization algorithms exploiting unitary constraints. *IEEE Transactions on Signal Processing*, *50*(3), 635–650.

Mirzaei, F. M., Kottas, D. G., & Roumeliotis, S. I. (2012). 3D LIDAR-camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *International Journal of Robotics Research*, *31*(4), 452–467.

Moghadam, P., Bosse, M., & Zlot, R. (2013). Line-based extrinsic calibration of range and image sensors. In *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany (pp. 3685–3691).

Napier, A., Corke, P., & Newman, P. (2013). Cross-calibration of push-broom 2D LIDARs and cameras in natural scenes. In *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany (pp. 3679–3684).

Ni, H., Lin, X., Ning, X., & Zhang, J. (2016). Edge detection and feature line tracing in 3D-point clouds by analyzing geometric properties of neighborhoods. *Remote Sensing*, *8*(9:710), 1–20.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. New York: Springer.

Pandey, G., McBride, J. R., & Eustice, R. M. (2011). Ford campus vision and lidar data set. *International Journal of Robotics Research*, *30*(13), 1543–1552.

Pandey, G., McBride, J. R., Savarese, S., & Eustice, R. M. (2015). Automatic extrinsic calibration of vision and lidar by maximizing mutual information. *Journal of Field Robotics*, *32*(5), 696–722.

Pascoe, G., Maddern, W., & Newman, P. (2015). Direct visual localisation and calibration for road vehicles in changing city environments. In *International Conference on Computer Vision Workshops*, Boston, Massachusetts (pp. 98–105).

Scaramuzza, D., Harati, A., & Siegwart, R. (2007). Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, California (pp. 4164–4169).

Sobel, I., & Feldman, G. (1968). *A 3x3 isotropic gradient operator for image processing*. Presented at the Stanford Artificial Intelligence Project.

Strom, J., Richardson, A., & Olson, E. (2010). Graph-based segmentation for colored 3D laser point clouds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan (pp. 2131–2136).

Tamas, L., & Kato, Z. (2013). Targetless calibration of a lidar-perspective camera pair. In *International Conference on Computer Vision Workshops*, Sydney, Australia (pp. 668–675).

Taylor, Z., & Nieto, J. (2012). A mutual information approach to automatic calibration of camera and lidar in natural environments. In *Australasian Conference on Robotics and Automation*, Wellington, New Zealand (pp. 1-8).

Taylor, Z., Nieto, J., & Johnson, D. (2015). Multi-modal sensor calibration using a gradient orientation measure. *Journal of Field Robotics*, *32*(5), 675–695.

The MathWorks Inc. (2017). *MATLAB and optimization toolbox release 2017b*. Natick, MA: The MathWorks Inc.

Unnikrishnan, R., & Hebert, M. (2005). *Fast extrinsic calibration of a laser rangefinder to a camera* (Technical Report No. CMU-RI-TR-05-09). Pittsburgh, PA: Carnegie Mellon University.

Wang, R., Ferrie, F. P., & Macfarlane, J. (2012). Automatic registration of mobile LiDAR and spherical panoramas. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Providence, Rhode Island (pp. 33–40).

Wei, J., Snider, J. M., Kim, J., Dolan, J. M., Rajkumar, R., & Litkouhi, B. (2013). Towards a viable autonomous driving research platform. In *Intelligent Vehicles Symposium*, Gold Coast City, Australia (pp. 763–770).

Weinmann, M., Jutzi, B., Hinz, S., & Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, *105*, 286–304.

Wolfe, P. (1969). Convergence conditions for ascent methods. *SIAM Review*, *11*(2), 226–235.

Xia, S., & Wang, R. (2017). A fast edge extraction method for mobile lidar point clouds. *IEEE Geoscience and Remote Sensing Letters*, *14*(8), 1288–1292.

Xu, L., Lu, C., Xu, Y., & Jia, J. (2011). Image smoothing via L_0 gradient minimization. *ACM Transactions on Graphics*, *30*(6), 174:1–174:12.

Zhang, J., & Singh, S. (2015). Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *IEEE International Conference on Robotics and Automation*, Seattle, Washington (pp. 2174–2181).

Zhang, Q., & Pless, R. (2004). Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan (pp. 2301–2306).

How to cite this article: Kang J, Doh NL. Automatic targetless camera-LIDAR calibration by aligning edge with Gaussian mixture model. *J Field Robotics*. 2020;37:158–179. https://doi.org/10.1002/rob.21893