2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

An Automatic LiDAR-Camera Extrinsic Calibration Method for Sparse Point Clouds Using Boundary Features

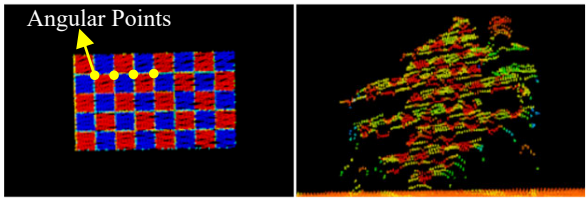
Tiancheng Gu, Minqian Wang, Libo Weng, Yanjing Lei and Fei Gao*

***Abstract*— Extrinsic calibration for LiDAR and camera using sparse point clouds can significantly reduce cost and improve efficiency. However, most current target-based methods are designed for dense point clouds and are less effective in sparse scenarios, while targetless methods primarily rely on environmental features. To address this limitation, a LiDAR–camera extrinsic calibration method for sparse point clouds is proposed in this paper. First, the proposed method extracts the complete checkerboard image via line-segment direction clustering and midpoint-to-normal projection. Second, a constructed theoretical checkerboard boundary point cloud is aligned to the scanned boundary point cloud using a proposed dimension-reduced, global-search and local-refinement (DGL) method. Third, coarse calibration is derived from the centroids of the checkerboard in images and aligned point clouds, followed by refinement through joint optimization of reprojection error and normal consistency error. Finally, experiments on simulated data achieve translation and rotation errors below 0.015 m and 0.3°, respectively. On a self-collected dataset, the method attains 90.9% mIoU between reprojected checkerboard regions and images, outperforming state-of-the-art methods under sparse point cloud conditions.**

I. INTRODUCTION

In recent years, LiDAR–camera fusion has attracted significant attention in autonomous driving [1] [2] [3], robotics [4] [5], and 3D reconstruction [6], and has become one of the research hotspots in the field of 3D vision. Accurate extrinsic calibration is a prerequisite for LiDAR–camera fusion. It aims to estimate the rigid transformation between the two sensors through mathematical modeling and optimization, thereby aligning their coordinate systems. Extrinsic calibration forms the basis for precisely projecting 3D point clouds onto 2D images. Consequently, obtaining high-precision and robust extrinsic parameters is a challenging task in LiDAR–camera fusion.

Current LiDAR–camera extrinsic calibration methods can be classified into two types: targetless methods and target-based methods. Targetless methods [7] [8] [9] [10] [11] exploit geometric or semantic features in natural scenes instead of artificial calibration objects. However, they are often sensitive to initialization and face difficulties in feature extraction and data association under repetitive textures or sparse point clouds. In addition, an increasing number of learning-based targetless methods [12] [13] [14] [15] [16] have been proposed. Although these approaches show strong potential in automation and accuracy, they rely heavily on large annotated datasets and may not generalize well to customized sensor setups or diverse environments. At present, widely used calibration techniques typically adopt target-based methods [17] [18] [19], which rely on dedicated calibration objects observed by both sensors. However, such methods are prone to failure due to the lack of sufficient 3D corner or edge features under sparse point cloud conditions, as illustrated in Fig. 1. Other works, such as Yan et al. [20], Huang et al. [21] and Tóth et al. [22] employ customized calibration targets, which increase both cost and complexity.



(a) Dense Point Cloud (b) Sparse Point Cloud

Fig. 1. Comparison of features between dense and sparse point clouds.

In summary, current methods suffer from difficulties in feature association, corner extraction, and other issues when dealing with sparse point clouds, which often result in reduced accuracy or even failure. Moreover, acquiring dense point clouds is costly and computationally demanding, especially for spinning LiDARs. To address these limitations, a LiDAR–camera extrinsic calibration method for sparse point clouds is proposed, which employs a tilted rigid checkerboard as the calibration target and follows a coarse-to-fine optimization strategy. An initial transformation is estimated from the centroids of corresponding checkerboard regions in point clouds and images, and a joint optimization framework is introduced to enable accurate automatic calibration even with imperfect initialization.

The main contributions of the proposed method are as follows:

- An automatic method for extracting complete checkerboard boundary based on line-segment direction clustering is proposed, which improves system automation and avoids errors introduced by manual extraction.
- By replacing scanned checkerboard boundary with theoretical boundary point cloud derived from the checkerboard dimensions, a dimension-reduced, global-search, and local-refinement (DGL) alignment approach is pro-

*The corresponding author of this paper.
The authors are with College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. {221124120158, 121122120025, wenglibo, leiyj, feig}@zjut.edu.cn

This work is being supported by the Zhejiang Provincial Science and Technology Planning Key Project of China under Grant No. 2026C01026 and 2026C02A3024, and the Hangzhou Science and Technology Planning Key Project of China under No. 2025SZD1A01.

posed for high-precision boundary alignment in sparse point cloud scenarios.
- Both boundary reprojection error and normal consistency error are incorporated in the optimization-based fine calibration phase, which aims to reduce errors caused by directional deviation of the point cloud.

II. RELATED WORK

Current methods of LiDAR–camera extrinsic calibration can be classified into target-based methods and targetless methods.

Target-based methods are the most established approach for LiDAR–camera extrinsic calibration. The key idea is to use specially designed calibration objects that are simultaneously observable by both sensors. Commonly used targets include checkerboards [23] and fiducial markers such as AprilTag [24] [25], which provide geometric features (e.g., corners and edges) for correspondence matching. Jeong et al. [26] improved board extraction by augmented points, but their method relies on the known distribution of checkerboard inner corners and is limited by sparse scans at long ranges. Su et al. [27] introduced a motion-based coarse calibration that is not applicable in static scenes. Hua et al. [17] optimized poses using edge constraints for more accurate rigid transformations, yet edge extraction remains sensitive to point cloud density and requires manual intervention in complex backgrounds. Other studies designed customized targets—such as boards with circular holes [20] [28] [29], transparent acrylic checkerboards [21], or spherical objects [22]—to meet specific needs; however, these are costly and complex to fabricate, and their manufacturing precision directly affects calibration accuracy.

Targetless methods eliminate the need for specific calibration objects by exploiting natural features in the environment. These approaches typically rely on geometric alignment or statistical measures between LiDAR and camera data. Chen et al. [9] proposed a plane-constrained bundle adjustment method that iteratively minimizes reprojection error using feature points on prominent scene planes. Borer et al. [10] introduced a method based on geometric mutual information, replacing the traditional intensity-to-intensity feature with depth-to-depth feature. Koide et al. [11] employed SuperGlue [30] for cross-modal 2D–3D matching, then refined the initial guess with direct LiDAR-camera registration based on the normalized information distance. However, many targetless methods rely on scene-dependent cues that may be unreliable or absent in textureless or repetitive environments, resulting in reduced accuracy compared to target-based approaches.

In recent years, learning-based methods that extract modality-invariant features have emerged as a form of targetless methods. Yuan et al. [12] proposed a network that integrates Riemannian geometry with a deep generative model to learn an implicit tolerance model. Zhu et al. [13] formulated calibration as a sequence prediction task with LSTMs for contextual modeling. Sun et al. [14] performed attention-based object-level matching with cascaded optimization, enabling target-free and initialization-free calibration. Lv et al. [15] developed an end-to-end self-calibration network with a cost-volume mechanism. Lee et al. [16] leveraged 4D correlation volumes with iterative updates to achieve cross-modal alignment. Ye et al. [31] embedded geometric constraints and trainable keypoint weighting into an end-to-end network. Zhu et al. [32] employed homogeneous multi-modality representation with local-global spatial awareness. Jing et al. [33] proposed a differentiable pose estimation module with probabilistic modeling to quantify cross-modal uncertainty. Despite their strong performance and automation, these methods face limitations: their generalization heavily depends on training datasets, often failing to transfer across environments or sensor configurations. Moreover, their reliance on large-scale computation and labeled data restricts deployment in lightweight systems.

The proposed method remains effective for sparse point clouds, performing robustly with low-resolution LiDAR or long-range setups. It also achieves reliable calibration in textureless or repetitive-structure scenarios and is applicable to both indoor and outdoor settings.

III. METHODOLOGY

A. Overview

The proposed method adopts a coarse-to-fine strategy with staged optimization. A checkerboard is tilted and placed at different positions within the overlapping LiDAR–camera field of view (FOV) to ensure that its boundaries are captured. The method consists of three modules: (1) data preprocessing, (2) coarse calibration, and (3) fine calibration. In preprocessing, planar checkerboard point clouds are obtained, and a checkerboard image extraction method based on line-segment direction clustering is proposed. In coarse calibration, theoretical boundary point clouds are aligned to the scanned ones, and paired centroids of the theoretical point clouds and checkerboard images are used to solve a Perspective-n-Points (PnP) problem for an initial guess. In fine calibration, boundary reprojection error and normal consistency error are jointly minimized to refine the result. The overall pipeline is shown in Fig. 2.

B. Data Preprocessing

1) Checkerboard point cloud extraction: Accurate calibration requires first extracting the checkerboard point cloud. Redundant points outside the checkerboard region are removed using spatial thresholds determined by the acquisition setup. A plane is then fitted to the filtered points to remove ground points. From the remaining non-ground points, a second plane fitting yields the checkerboard plane α, and points within a small distance to α are preserved as the checkerboard point cloud P_b. Due to inherent LiDAR noise and environmental factors, P_b does not form an exact plane and contains noise. Therefore, all points in P_b are projected onto α using ray projection, as defined in (1), where A, B, C, D are the parameters of α, producing the 3D planar checkerboard point cloud P_r.

$$P_r = \left\{ -\frac{D}{Ax + By + Cz} \cdot (x, y, z) \;\middle|\; (x, y, z) \in P_b \right\} \quad (1)$$

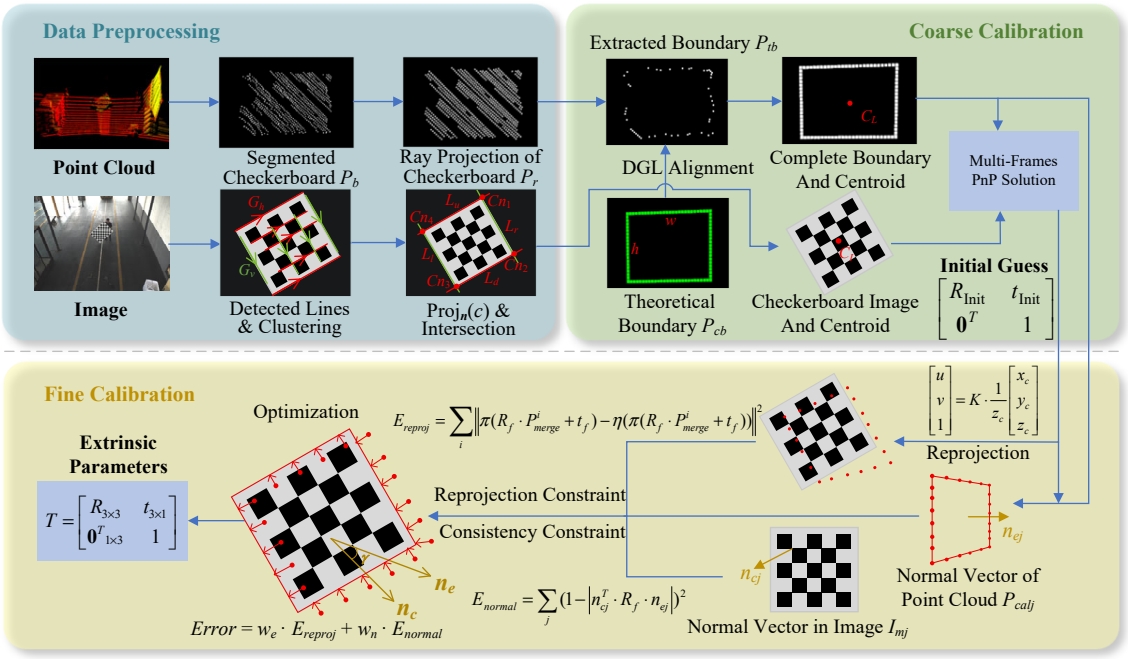


Fig. 2. Overview of the proposed method.

2) Checkerboard image extraction: Most checkerboard extraction methods rely on corner detection, which often results in incomplete extraction. Thus, a method based on line-segment direction clustering is proposed to extract the complete checkerboard boundary from the image.

First, the bounding box of the checkerboard is obtained and expanded outward to generate a new bounding box. Line detection is then performed within this region to extract a set of candidate boundary line segments $\{L_i\}_{i=1}^{k}$, as shown in Fig. 3(a). The unit direction vector v_i of each line segment L_i is then computed.

Based on v_i, $\{L_i\}_{i=1}^{n}$ is clustered into two groups, G_v and G_h, corresponding to the two principal orientations of the checkerboard, as shown in Fig. 3(b). When a group of approximately parallel lines exists in the image, the projections of their centroids along the normal vector direction can accurately reflect their relative positions. Taking G_h as an example, the first line segment in G_h and its normal vector n_l is computed according to (2).

$$n_l = (-v_1^{(y)}, v_1^{(x)}) \tag{2}$$

For each line segment in G_h, the midpoint c_i is determined, and their projection values onto n_l are obtained as CP according to (3), where j denotes the number of line segments in the cluster.

$$CP = \begin{bmatrix} c_1 \cdot n_l^T & c_2 \cdot n_l^T & \cdots & c_j \cdot n_l^T \end{bmatrix}^T \tag{3}$$

The line segments corresponding to the maximum and minimum CP values are selected as the outermost line segments of G_h, denoted as the upper and lower boundaries L_u and L_d, respectively, as shown in Fig. 3(c). Likewise, the right and left boundaries L_r and L_l are obtained from G_v. The intersections of adjacent boundaries determine four checkerboard corners Cn_1–Cn_4, as illustrated in Fig. 3(d). Mapping these corners back to the original image, the quadrilateral region they form is extracted as the checkerboard image I_m, with its contour points denoted as $Contour$.

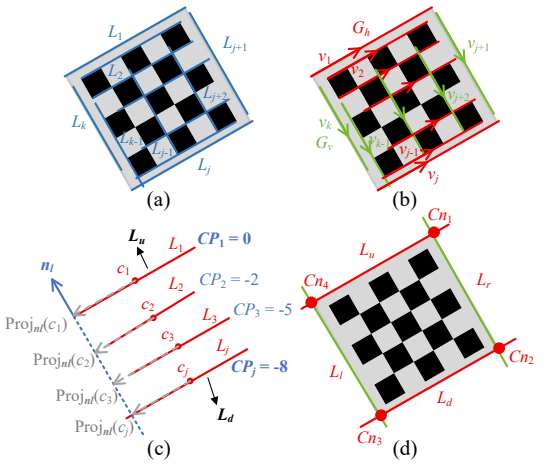


Fig. 3. Extraction of the complete checkerboard image. (a) Extracted line segment set $\{L_i\}_{i=1}^{k}$. (b) Clusters G_v and G_h obtained by line-segment direction clustering. (c) Two outermost line segments extracted in G_h based on midpoint-to-normal projection. (d) Boundary lines and corners of the checkerboard.

C. Coarse Calibration

1) Theoretical boundary alignment: Boundary extraction algorithm provides the checkerboard boundary point cloud P_{tb}, as shown in Fig.4(a). Due to sparsity, some segments may be missing or misidentified, causing calibration errors. To address this, a theoretical boundary P_{cb} based on the

checkerboard size $w \times h$ is constructed, as shown in Fig.4(b), and aligned to P_{tb} using the proposed DGL (Dimension-reduced, Global-search, Local-refinement) method.

The method computes covariance matrices of P_{cb} and P_{tb}, and performs eigen-decomposition to obtain the orthogonal bases V_{cb} and V_{tb}. Using V_{tb} as reference, the transformation matrix mapping P_{cb} onto the target plane α is obtained through principal direction alignment and centroid matching, as given in (4), where μ_{cb} and μ_{tb} denote the centroids of P_{cb} and P_{tb}, respectively. The resulting transformation is then applied to P_{cb}.

$$T_{\text{Init}} = \begin{bmatrix} V_{tb}V_{cb}^T & \mu_{tb} - V_{tb}V_{cb}^T\mu_{cb} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{4}$$

After this step, both point clouds are further rotated onto a plane parallel to the XOY plane, reducing the alignment parameters to three (Dimension-reduced): rotation angle θ around the Z-axis and translations t_x, t_y. A global search strategy finds an approximately optimal solution (Global-search), which is then refined via local optimization (Local-refinement). The alignment optimization, defined in (5), minimizes the mean squared distance from each transformed source point to its nearest neighbor in the target point cloud P_{tb}, as illustrated in Fig. 4(c). Here, R_Z and t_Z denote the rotation matrix and translation vector, respectively; p_i denotes a point in the source point cloud; $\min_{q_j \in P_{tb}}$ represents the point q_j in the target point cloud, found as the nearest neighbor to the transformed p_i.

$$\begin{aligned} &\text{Find} \quad \theta, t_x, t_y \\ &\min \quad f = \frac{1}{n}\sum_{i=1}^{n} \min_{q_j \in P_{tb}} \|R_Z(\theta)\cdot p_i + t_Z(t_x,t_y) - q_j\|^2 \\ &\text{s.t.} \quad R_Z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; t_Z(t_x,t_y) = \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix}, \\ &\quad -90^\circ \le \theta \le 90^\circ \end{aligned} \tag{5}$$

This method reduces the optimization parameters from six (three for rotation and three for translation) to three, as shown in Fig. 4(d), thereby improving efficiency and reducing error. After obtaining the optimal parameters θ', t'_x and t'_y, the transformation matrix T_Z is constructed according to (6).

$$T_Z = \begin{bmatrix} R_Z(\theta') & t_Z(t'_x, t'_y) \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{6}$$

The theoretical point cloud is then transformed by T_Z and rotated back onto the plane α, yielding the aligned theoretical boundary P_{eb}. This boundary represents the actual position of the checkerboard in 3D space and is subsequently used for extrinsic calibration.

2) Initial extrinsic estimation: Since the checkerboard points are coplanar, a single point cloud–image pair cannot provide sufficient constraints for stable PnP estimation. Thus, the board is scanned at least four times at different positions within the FOV while keeping the LiDAR and camera fixed. From the s ($s \ge 4$) pairs, the theoretical boundary point cloud set $PEB = \{P_{eb1}, \ldots, P_{ebs}\}$ and corresponding

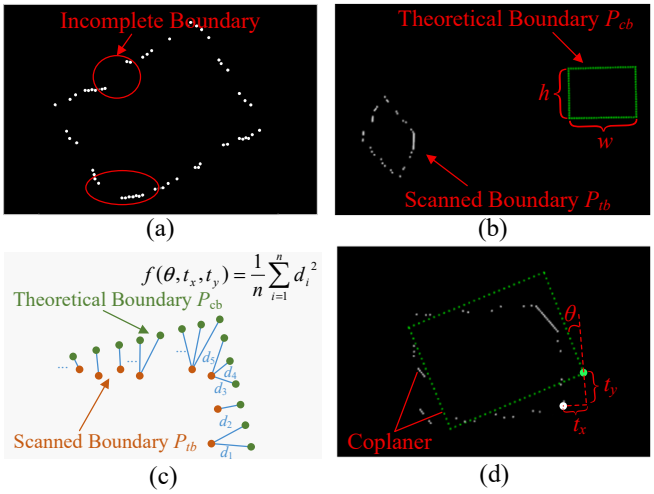


Fig. 4. Theoretical boundary alignment. (a) Extracted incomplete boundary. (b) Scanned boundary and theoretical boundary. (c) Error function in alignment optimization. (d) Dimensionality reduction of optimization parameters.

image set $IM = \{I_{m1}, \ldots, I_{ms}\}$ are obtained. For each pair (P_{ebi}, I_{mi}), the checkerboard centroid in the point cloud (C_{Li}) and the centroid in the image (C_{Ii}) are extracted, forming centroid sets CL and CI. Solving PnP with CL and CI yields the initial extrinsic parameters $(rvec, tvec)$.

D. Fine Calibration

1) Checkerboard normal vector calculation: Using the coarse calibration parameters $rvec$ and $tvec$, each point cloud in PEB is transformed into the camera coordinate system according to (7), yielding $PCAL = P_{cal1}, \ldots, P_{cals}$. Here, Rodrigues$(\cdot)$ converts $rvec$ into a rotation matrix, and P_{cali} denotes the homogeneous coordinates in the camera system.

$$P_{cali} = \begin{bmatrix} \text{Rodrigues}(rvec) & tvec \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot P_{ebi} \tag{7}$$

During the transformation, the orientation of checkerboard boundary point cloud may deviate, and further tilting can occur during fine calibration. To address this, the orientation of P_{cali} in the camera coordinate system is represented by a normal vector n_{ei}, while the orientation of the corresponding checkerboard in the image is represented by a normal vector n_{ci}, as illustrated in Fig. 5. The consistency between n_{ei} and n_{ci} is then constrained in the subsequent optimization.

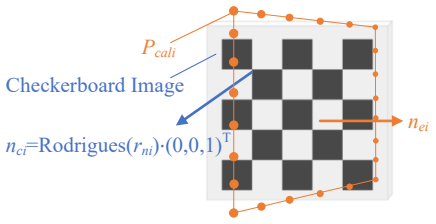


Fig. 5. Tilt of the boundary point cloud.

The boundary point cloud normal n_{ei} is obtained directly. For the checkerboard, the local world coordinate system is

defined with the X-axis pointing right, the Y-axis downward, and the Z-axis perpendicular to the plane toward the camera, giving the normal $n_w = (0, 0, 1)^T$. Using detected inner corners and their 3D world coordinates, the PnP algorithm estimates the rotation r_{ni} from the world to the camera coordinates, which is then applied to n_w to obtain n_{ci} as in (8). Since the sign ambiguity of the normals does not affect the subsequent optimization, no orientation constraint is imposed on n_{ei} and n_{ci}.

$$n_{ci} = \text{Rodrigues}(r_{ni}) \cdot n_w \tag{8}$$

2) Extrinsic refinement: The coarse calibration provides a reasonably reliable initial solution; therefore, the extrinsic parameters can be further refined around this solution using local optimization methods.

Quaternion representation. In this stage, rotation is represented by quaternions $q = [w, x, y, z]$ instead of Euler angles (roll, pitch, yaw) to avoid strong nonlinearity, gradient instability, and gimbal lock. A unit quaternion $q = w + xi + yj + zk$ ($w^2 + x^2 + y^2 + z^2 = 1$) corresponds to a rotation matrix R_f as in (9). Since the manifold of unit quaternions is nonlinear, rotation increments are defined in Lie algebra and mapped to quaternions via the exponential map for stable updates.

$$R_f = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{bmatrix} \tag{9}$$

Joint optimization. Observations from multiple frames provide richer geometric constraints, enabling joint optimization of extrinsic parameters and compensating for degraded frames. Accordingly, all point clouds P_{cali} and 2D contours $Contour_i$ are merged into P_{merge} and C_{merge}, respectively, for joint refinement.

Error function. For each point (x_c, y_c, z_c) in P_{merge}, the reprojection function π maps it to image coordinates (u, v) using the known intrinsic matrix K, as shown in (10).

$$\begin{bmatrix} u & v & 1 \end{bmatrix}^T = K \cdot \frac{1}{z_c} \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T \tag{10}$$

To align theoretical boundary points to the checkerboard contours, the reprojection error E_{reproj} is the sum of squared distances between each reprojected point and its nearest neighbor in the merged contour set C_{merge}, as defined in (11), where t_f denotes the translation vector and $\eta(\cdot)$ returns the closest contour point in C_{merge}.

$$\sum_i \left\| \pi(R_f \cdot P^i_{merge} + t_f) - \eta\left(\pi(R_f \cdot P^i_{merge} + t_f)\right) \right\|^2 \tag{11}$$

Using only E_{reproj} may cause deviation in the orientation of the boundary point cloud. To enforce consistency with the true checkerboard orientation, a normal consistency error E_{normal} is introduced in (12). Specifically, parallelism between the n_{cj} and n_{ej} is quantified using the absolute value of their dot product, thereby avoiding sensitivity to sign ambiguity in normal directions.

$$E_{normal} = \sum_j \left(1 - \left| n^T_{cj} \cdot R_f \cdot n_{ej} \right|\right)^2 \tag{12}$$

Overall, the combination of E_{reproj} and E_{normal} mitigates local minima. The complete error function E is defined in (13), where w_e and w_n are weighting factors.

$$E = w_e \cdot E_{reproj} + w_n \cdot E_{normal} \tag{13}$$

Since the two error terms differ significantly in scale, a dynamic weighting scheme based on root mean square (RMS) is adopted. w_e is fixed to 1.0, while w_n is adaptively updated after each iteration as shown in (14), ensuring balanced contributions and preventing dominance of a single term.

$$w_n = \frac{\text{RMS}(E_{reproj})}{\text{RMS}(E_{normal})} \tag{14}$$

After minimizing E, the optimal rotation q' and translation t' are obtained. The rotation matrix R' is derived from q' via (9). Combined with the coarse calibration result, the refined transformation T' is shown in (15), representing the final extrinsic parameters.

$$T' = \begin{bmatrix} R' & t' \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot \begin{bmatrix} \text{Rodrigues}(rvec) & tvec \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{15}$$

IV. EXPERIMENTS

A. Setup

A hybrid solid-state LiDAR (FOV $180° \times 40°$) and an area scan camera (FOV $56.5° \times 44.6°$) were mounted rigidly on a mast arm, with the camera positioned above the LiDAR. The checkerboard (1.35 m$\times 1.01$ m) was fixed on the ground at an inclined pose. LiDAR–camera synchronization at the nanosecond level was achieved via a network switch. Example data are shown in Fig. 6(a) and Fig. 6(b). The captured images were preprocessed with distortion correction. To quantitatively evaluate performance against ground truth, a simulated dataset generated by H. Huang et al. [18] using the Gazebo simulator is employed. Dense point clouds in the dataset are randomly downsampled to retain 5% of the points, yielding relatively sparse point clouds. Example data are shown in Fig. 6(c) and Fig. 6(d). All experiments were conducted on a computational platform featuring an Intel Core i7-13650HX processor.

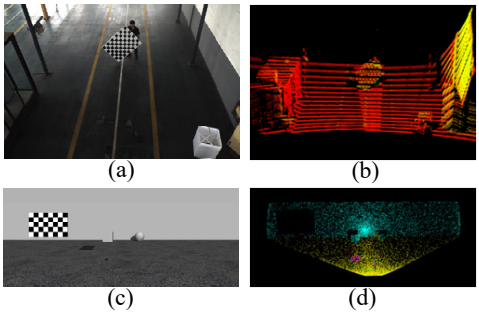


Fig. 6. Datasets. (a) and (b) are real-world data. (c) and (d) are simulated data.

B. Error Definition

1) Simulated dataset: For the simulated dataset, the ground-truth extrinsic parameters are directly available. The translation error e_t is measured by the L2 norm, and the rotation error e_R is measured by the axis-angle error, as defined in (16), where t_{est} and t_{gt} denote the estimated and ground-truth translation vectors, respectively, and R_{est} and R_{gt} denote the estimated and ground-truth rotation matrices, respectively.

$$\begin{cases} e_t = \|t_{est} - t_{gt}\|_2 \\ e_R = \arccos\left(\frac{\text{trace}(R_{gt}^T R_{est}) - 1}{2}\right) \end{cases} \tag{16}$$

2) Real-world dataset: For the real-world dataset, ground-truth extrinsic parameters are unavailable. To evaluate the calibration, the reprojected checkerboard boundary points are first sorted by polar angle and connected to form a closed polygon. This polygon is then compared with the corresponding board region in the image. The mean Intersection over Union (mIoU) over multiple frames is used as the evaluation metric, as defined in (17), where T is the number of point cloud–image samples, E_t denotes the polygonal region of the t-th reprojected boundary, and G_t denotes the checkerboard region in the t-th image.

$$\text{mIoU} = \frac{1}{T}\sum_{t=1}^{T}\frac{|E_t \cap G_t|}{|E_t \cup G_t|} \tag{17}$$

C. Ablation Study on Normal Consistency

To evaluate the effect of the normal consistency term in fine calibration, two error functions are compared: one combines boundary reprojection error E_{reproj} and normal consistency error E_{normal}, while the other uses only E_{reproj}. Experiments on the simulated dataset quantify translation and rotation errors across different sets of point cloud–image pairs. The results are shown in Fig. 7.

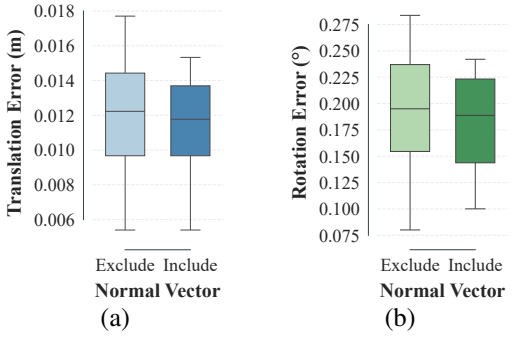


Fig. 7. Impact of normal consistency on calibration error. (a) Impact on translation error. (b) Impact on rotation error.

The results indicate that including the normal constraint improves convergence to a more plausible optimum. Without it, the ranges of translation and rotation errors are $\Delta e_t = 0.0123$ m and $\Delta e_R = 0.2036°$, respectively; with it, they reduce to $\Delta e_t = 0.0099$ m and $\Delta e_R = 0.1419°$, demonstrating higher accuracy and more stable rotation estimation, particularly in rotation estimation.

D. Effect of Alignment Method

To assess the effectiveness of the proposed DGL method, three optimization strategies are compared during the theoretical boundary alignment: (1) PSO-based; (2) LM-based; and (3) DGL with reduced parameters combining PSO for global search and LM for local refinement. Each strategy performs alignment using the same set of point cloud–image pairs. The swarm size in PSO is varied from 20 to 200 in steps of 20, with 10 runs per setting to compute calibration errors. Since LM is deterministic, it is executed only once.

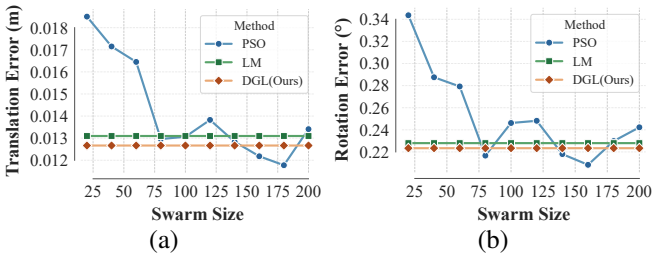


Fig. 8. Impact of different alignment methods on calibration error. (a) Impact on translation error. (b) Impact on rotation error.

As shown in Fig. 8, the DGL method outperforms PSO and LM alone in calibration accuracy. As the swarm size increases, the performance of PSO gradually approaches that of DGL and occasionally exceeds it, but it exhibits larger fluctuations and higher computational cost. Thus, the DGL alignment method enhances accuracy, efficiency, and robustness against local minima.

E. Performance Comparison

The proposed method is compared with several methods: checkerboard-based methods GTSCalib [18] and ACLC [19], a mutual-information-based method DVLC [11], a geometry-consistency-based method CalibAnything [34], and a learning-based method MRCNet [35].

1) Simulated dataset: On the simulated dataset, Euler angles are unreliable because a ground-truth pitch of $-90°$ causes gimbal lock. Rotation is thus evaluated using the axis–angle error e_R.

TABLE I
CALIBRATION PERFORMANCE OF DIFFERENT METHODS ON THE SIMULATED DATASET

Methods	Translation (m)			e_t (m)	e_R (°)
	X	Y	Z		
Ground Truth	0.0000	0.0500	0.0000	0.0000	0.0000
GTSCalib [18]			Failed		
ACLC [19]			Failed		
DVLC [11]	0.1512	0.0860	-0.2656	0.3077	0.3403
CalibAnything [34]	0.2466	0.2433	1.1550	1.1967	13.2562
MRCNet [35]	-1.7712	1.9376	2.2753	3.4463	12.5309
Ours	**0.0097**	**0.0528**	**-0.0076**	**0.0127**	**0.2233**

As shown in Table I, our method outperforms all others, achieving e_t below 1.5 cm and e_R below $0.3°$. GTSCalib and ACLC fail due to insufficient point cloud density for

3D corner extraction. CalibAnything and MRCNet are able to estimate extrinsics but with larger errors. CalibAnything relies on Segment Anything Model (SAM) for segmentation, but segmentation results on this dataset are suboptimal, and the sparse point clouds provide insufficient constraints for optimization, leading to poor accuracy.

2) Real-world dataset: On the real-world dataset, performance is evaluated using mIoU. Although the ideal mIoU is 1.0, it is much lower in practice due to the sparsity and incompleteness of boundary point clouds, as illustrated in Fig. 9. Nevertheless, mIoU still remains a fair and consistent metric for method comparison on the same point cloud–image pairs.

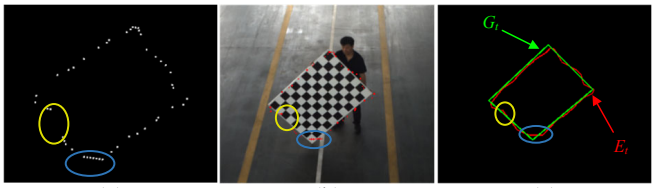


Fig. 9. Illustration of incomplete closed polygons in IoU computation. (a) Incomplete point cloud boundary. (b) Reprojection of the incomplete boundary. (c) Incomplete closed polygon.

As shown in Table II, our method attains the highest mIoU. DVLC performs poorly due to large sparse areas in the point clouds, which introduce substantial errors in extrinsic estimation. MRCNet, trained on KITTI datasets, works well in outdoor driving scenes but struggles to generalize to our dataset, which lacks similar scenes and objects, leading to frequent mismatches.

TABLE II

CALIBRATION PERFORMANCE OF DIFFERENT METHODS ON THE REAL-WORLD DATASET

Methods	Translation (m)			Rotation (°)			mIoU
	X	Y	Z	Roll	Pitch	Yaw	
GTSCalib [32]	Failed						
ACLC [33]	Failed						
DVLC [11]	0.411	1.124	0.135	97.663	-1.540	-3.605	0.53259
CalibAnything [34]	-0.009	0.121	0.122	89.849	1.014	-0.120	0.89565
MRCNet [35]	-0.402	-0.135	1.446	83.056	3.864	4.343	0.26596
Ours	0.090	0.146	0.120	90.099	0.106	0.580	**0.90602**

F. Performance with limited samples

Collecting many pairs of LiDAR–camera data is time-consuming. Thus, only a limited number of samples are used: 4–7 point cloud–image pairs were randomly selected for calibration. For fairness, mIoU was computed on the same set of pairs, and the selection and calibration were repeated multiple times. Results are shown in Fig. 10.

With 4–7 sample pairs, the method achieves stable performance: on the simulated dataset, translation errors remain within 0.008–0.015 m and rotation errors within 0.1°–0.275°; on the real-world dataset, mIoU stays between 0.896 and 0.910. Distortions, noise and blurring in real imaging, which vary across regions and viewpoints, make additional samples beneficial, explaining the upward trend in Fig. 10(c). Notably, even with only four pairs, the method yields reliable extrinsic results, demonstrating strong generalization in data-limited scenarios. Utilizing parallel PSO with a swarm size of 20, the proposed method requires a total time of 8.68 s to complete the pipeline for four point cloud–image pairs.

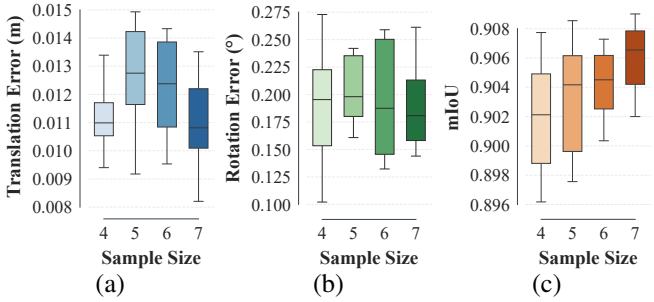


Fig. 10. Calibration performance versus sample size. (a) Translation error on the simulated dataset. (b) Rotation error on the simulated dataset. (c) mIoU on the real-world dataset.

G. Reprojection visualization

The point clouds were reprojected onto the images using the estimated extrinsics, with ground points removed for clarity. As shown in Fig. 11, the reprojected points align well with objects in the images.

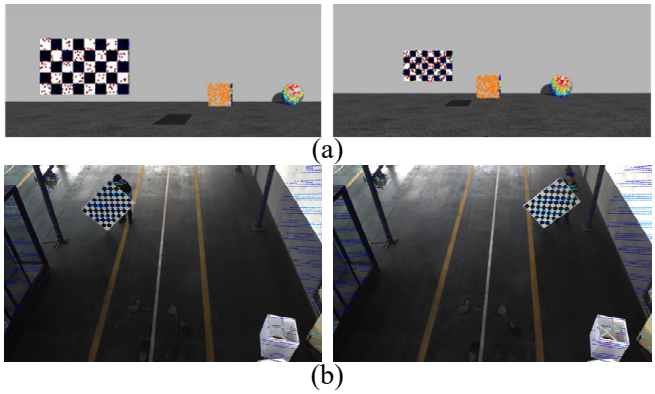


Fig. 11. Reprojection visualization. (a) Reprojection on the simulated dataset. (b) Reprojection on the real-world dataset.

V. CONCLUSION

In this paper, a LiDAR–camera extrinsic calibration method for sparse point clouds using boundary features is proposed. Complete checkerboard images are automatically extracted, and the scanned boundaries are replaced with theoretical boundaries using the proposed DGL alignment method. Boundary reprojection and normal consistency errors are incorporated into the refinement stage. Future work will focus on enhancing real-time performance and extending the method to checkerboards of unknown size.

REFERENCES

[1] Y. Qin, C. Wang, Z. Kang, N. Ma, Z. Li, and R. Zhang, "Supfusion: Supervised lidar-camera fusion for 3d object detection," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 21 957–21 967.

[2] Y. Li, A. W. Yu, T. Meng, B. Caine, J. Ngiam, D. Peng, J. Shen, Y. Lu, D. Zhou, Q. V. Le, A. Yuille, and M. Tan, "Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 17 161–17 170.

[3] L. Zhang, X. Li, K. Tang, Y. Jiang, L. Yang, Y. Zhang, and X. Chen, "Fs-net: Lidar-camera fusion with matched scale for 3d object detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 12 154–12 165, 2023.

[4] Z. Lin, Z. Gao, B. M. Chen, J. Chen, and C. Li, "Accurate lidar-camera fused odometry and rgb-colored mapping," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2495–2502, 2024.

[5] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "Camvox: A low-cost and accurate lidar-assisted visual slam system," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5049–5055.

[6] W. Zhen, Y. Hu, J. Liu, and S. Scherer, "A joint optimization approach of lidar-camera fusion for accurate dense 3-d reconstructions," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3585–3592, 2019.

[7] Z. Bai, G. Jiang, and A. Xu, "Lidar-camera calibration using line correspondences," *Sensors*, vol. 20, no. 21, 2020.

[8] N. Ou, H. Cai, and J. Wang, "Targetless lidar-camera calibration via cross-modality structure consistency," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 2636–2648, 2024.

[9] F. Chen, L. Li, S. Zhang, J. Wu, and L. Wang, "Pbacalib: Targetless extrinsic calibration for high-resolution lidar-camera system based on plane-constrained bundle adjustment," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 304–311, 2023.

[10] J. Borer, J. Tschirner, F. Ölsner, and S. Milz, "From chaos to calibration: A geometric mutual information approach to target-free camera lidar extrinsic calibration," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 8394–8403.

[11] K. Koide, S. Oishi, M. Yokozuka, and A. Banno, "General, single-shot, target-less, and automatic lidar-camera extrinsic calibration toolbox," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 301–11 307.

[12] K. Yuan, Z. Guo, and Z. J. Wang, "Rggnet: Tolerance aware lidar-camera online calibration with geometric deep learning and generative model," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6956–6963, 2020.

[13] J. Zhu, J. Xue, and P. Zhang, "Calibdepth: Unifying depth map representation for iterative lidar-camera online calibration," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 726–733.

[14] Y. Sun, J. Li, Y. Wang, X. Xu, X. Yang, and Z. Sun, "Atop: An attention-to-optimization approach for automatic lidar-camera calibration via cross-modal object matching," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 696–708, 2023.

[15] X. Lv, B. Wang, Z. Dou, D. Ye, and S. Wang, "Lccnet: Lidar and camera self-calibration using cost volume network," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 2888–2895.

[16] Y.-C. Lee and K.-W. Chen, "Lccraft: Lidar and camera calibration using recurrent all-pairs field transforms without precise initial guess," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 16 669–16 675.

[17] Y. Hua, Q. Liu, T. Jiang, J. Zhang, W. Xu, and Y. Tian, "Accurate lidar–camera calibration using feature edges," *Image Vision Comput.*, vol. 155, no. C, Apr. 2025.

[18] H. Huang, M. Zhang, L. Li, J. Hu, and H. Wang, "Gtscalib: Generalized target segmentation for target-based extrinsic calibration of non-repetitive scanning lidar and camera," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 3648–3660, 2025.

[19] J. Cui, J. Niu, Y. He, D. Liu, and Z. Ouyang, "Aclc: Automatic calibration for nonrepetitive scanning lidar-camera system based on point cloud noise optimization," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–14, 2024.

[20] G. Yan, F. He, C. Shi, P. Wei, X. Cai, and Y. Li, "Joint camera intrinsic and lidar-camera extrinsic calibration," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 446–11 452.

[21] Z. Huang, X. Zhang, A. Garcia, and X. Huang, "A novel, efficient and accurate method for lidar camera calibration," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 14 513–14 519.

[22] T. Tóth, Z. Pusztai, and L. Hajder, "Automatic lidar-camera calibration of extrinsic parameters using a spherical target," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8580–8586.

[23] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2301–2306 vol.3.

[24] Y. Xie, R. Shao, P. Guli, B. Li, and L. Wang, "Infrastructure based calibration of a multi-camera and multi-lidar system using apriltags," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 605–610.

[25] L. Grammatikopoulos, A. Papanagnou, A. Venianakis, I. Kalisperakis, and C. Stentoumis, "An effective camera-to-lidar spatiotemporal calibration based on a simple calibration target," *Sensors*, vol. 22, no. 15, 2022.

[26] S. Jeong, S. Kim, J. Kim, and M. Kim, "O3 lidar–camera calibration: One-shot, one-target and overcoming lidar limitations," *IEEE Sensors Journal*, vol. 24, no. 11, pp. 18 659–18 671, 2024.

[27] Y. Su, Y. Ding, J. Yang, and H. Kong, "A two-step approach to lidar-camera calibration," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 6834–6841.

[28] J. Domhof, J. F. Kooij, and D. M. Gavrila, "An extrinsic calibration tool for radar, camera and lidar," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8107–8113.

[29] J. Beltrán, C. Guindel, A. de la Escalera, and F. García, "Automatic extrinsic calibration method for lidar and camera sensor setups," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 677–17 689, 2022.

[30] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4937–4946.

[31] C. Ye, H. Pan, and H. Gao, "Keypoint-based lidar-camera online calibration with robust geometric network," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

[32] A. Zhu, Y. Xiao, C. Liu, M. Tan, and Z. Cao, "Lightweight lidar-camera alignment with homogeneous local-global aware representation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 15 922–15 933, 2024.

[33] X. Jing, X. Ding, R. Xiong, H. Deng, and Y. Wang, "Dxq-net: Differentiable lidar-camera extrinsic calibration using quality-aware flow," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 6235–6241.

[34] Z. Luo, G. Yan, X. Cai, and B. Shi, "Zero-training lidar-camera extrinsic calibration method using segment anything model," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 14 472–14 478.

[35] H. Wang, Z. Wang, G. Yu, S. Yang, and Y. Yang, "Mrcnet: Multiresolution lidar-camera calibration using optical center distance loss network," *IEEE Sensors Journal*, vol. 24, no. 12, pp. 19 661–19 672, 2024.