Robust LiDAR-Camera Calibration with 2D Gaussian Splatting

Shuyi Zhou[1], Shuxiang Xie[1], Ryoichi Ishikawa[1], Takeshi Oishi[1]

***Abstract*— LiDAR-camera systems have become increasingly popular in robotics recently. A critical and initial step in integrating the LiDAR and camera data is the calibration of the LiDAR-camera system. Most existing calibration methods rely on auxiliary target objects, which often involve complex manual operations, whereas targetless methods have yet to achieve practical effectiveness. Recognizing that 2D Gaussian Splatting (2DGS) can reconstruct geometric information from camera image sequences, we propose a calibration method that estimates LiDAR-camera extrinsic parameters using geometric constraints. The proposed method begins by reconstructing colorless 2DGS using LiDAR point clouds. Subsequently, we update the colors of the Gaussian splats by minimizing the photometric loss. The extrinsic parameters are optimized during this process. Additionally, we address the limitations of the photometric loss by incorporating the reprojection and triangulation losses, thereby enhancing the calibration robustness and accuracy.**

I. INTRODUCTION

LiDAR-camera fusion plays a critical role in autonomous driving and robotics. By integrating accurate depth measurements from LiDAR with dense optical scans provided by cameras, we can develop robust solutions for various tasks, including object detection [1], simultaneous localization and mapping (SLAM) [2], and 3D reconstruction [3]. However, for effective sensor fusion, it is crucial to represent data from different sensors in a unified coordinate system.

Sensor calibration is an important preliminary step in integrating measurements from multiple sensors. The extrinsic calibration process attempts to determine the relative pose, encompassing translation and rotation between the sensors. Calibration among different sensors is typically challenging caused by variations in the density, sensor modalities, field of view, and resolution. Therefore, most traditional LiDAR-camera calibration methods use auxiliary calibration target objects, such as textured plane objects [4], [5], [6], [7]. However, these methods can be complex to set up and require laborious manual operations.

In the past decade, targetless calibration methods have emerged, beginning with techniques that use appearance correspondences, such as edges [8], [9] and intensities [10], [11] between LiDAR and camera frames. These methods have been followed by approaches that leverage deep learning techniques [12], [13]. Geometric constraints have also proven effective [14], [15], particularly with dense 3D reconstruction from images [16]. Recent advances in 3D reconstruction methods, such as neural radiance fields (NeRF) [17] and Gaussian splattings (GS) [18], have further refined this approach [19], [20]. These studies solve the domain gap between LiDAR and camera by aligning the geometry information measured from both sensors. However, the robustness of these methods depends on the differentiability of the underlying representation, and their accuracy is influenced by the rendering quality; thus, further improvement are required.

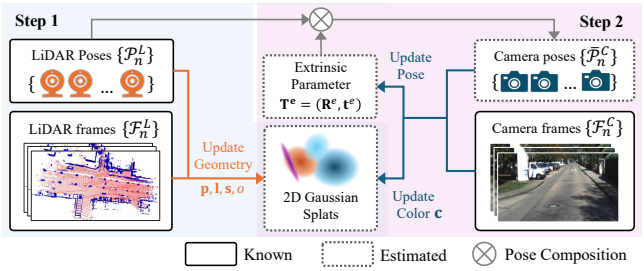


Fig. 1. Overview of the proposed method. The proposed method uses LiDAR frames to reconstruct the geometric properties of 2D Gaussian splatting and optimizes the LiDAR-camera extrinsic parameters while updating the colors of the 2D Gaussian splats.

Thus, we propose a LiDAR-camera calibration method to enhance robustness and accuracy using 2D Gaussian Splatting (2DGS), which leverages differentiability and high-quality rendering [21]. The proposed method is based on the concept of Implicit Neural Fusion (INF) [19], which aligns 3D scenes constructed by LiDAR scans with the geometry information inferred from the camera images. The proposed method begins by applying 2DGS to the LiDAR frames and then refines the Gaussian colors through photometric loss backpropagation from the camera images (Fig. 1), thereby enabling the simultaneous optimization of the LiDAR-camera extrinsic parameters. Meanwhile, we identified certain limitations in relying exclusively on photometric loss; thus, we introduced reprojection and triangulation loss terms to overcome these issues.

Our primary contributions are summarized as follows:

- We propose a targetless LiDAR-camera calibration method that consolidates various geometric constraints based solely on the 2DGS representation.
- We provide a mathematical analysis of the limitations inherent to the 2DGS for calibration and propose corresponding loss functions to address these limitations.
- We propose a depth weight uncertainty during the 2DGS reconstruction process to enhance the accuracy.

We conduct experiments using the KITTI odometry dataset [22], to demonstrate the robustness and accuracy of the proposed method.

This work was supported by JSPS KAKENHI Grant Numbers JP24K21173 and JP24H00351.

[1]The authors are with The Institute of Industrial Science, The University of Tokyo, Japan. Emails: {zhoushuyi495, shxxie, ishikawa, oishi}@cvl.iis.u-tokyo.ac.jp

II. RELATED WORK

This section briefly reviews LiDAR-camera calibration methods, 3D and 2D Gaussian splatting techniques, and simultaneous pose estimation algorithms used in 3D reconstruction process.

A. Targetless LiDAR-camera Calibration Methods

Conventional appearance-based methods estimate relative poses using corresponding visual features. The most practical and widely used methods [8], [9], [23], [24] align image edges with geometric edges in LiDAR data. [10] and [11] focused on maximizing the mutual information of intensity values between camera images and projected LiDAR points. In some studies [12], [13], [25], [26], neural networks were employed for feature matching. The above methods have certain limitations: the domain gaps between the 2D and 3D features reduce the accuracy and robustness, neural network (NN)-based methods require prior knowledge, and the capability to generalize NN-based methods remains unclear.

Motion-based methods [27], [14] rely on hand-eye calibration and estimate the extrinsic parameters by aligning the motion patterns of both sensors. These approaches require each sensor to independently estimate the relative poses across multiple frames. Additionally, motion-based methods depend on particular sensor movements to solve linear systems, which can be a significant limitation for certain robotic systems.

To address these limitations, geometry-based methods have been increasingly investigated [28], [15]. These approaches minimize the reprojection error to align LiDAR data with the geometry of multiview stereo cameras. Recent techniques have incorporated LiDAR-camera calibration within the process of 3D reconstruction [19], [29] using NeRF [17]. These methods bridge the domain gap by using the geometric consistency among sensors; however, their performance is highly dependent on the quality of the 3D reconstruction. The proposed method aligns with this line of research and employs 2DGS for scene representation.

B. Pose Estimation with Differentiable 3D Representations

3D reconstruction methods based on differentiable representations, such as NeRF, can estimate or improve camera poses in the optimization process. If we have sufficient input images, we can determine the shape and camera poses simultaneously according to the Structure from Motion (SfM) theorem [30]. This theorem can also be applied to NeRF; thus, there are various methods to estimate camera poses [31], [32], [33]. In addition, by leveraging consistent geometric information across sensors, some studies have achieved extrinsic calibration between LiDAR and cameras [19], [29], [34]. However, NeRF often faces challenges related to insufficient reconstruction quality, resulting in low calibration accuracy.

3DGS and the following 2DGS overcome these limitations by using explicit 3D Gaussian splats for scene representation. While Gaussian splatting is not globally continuous like implicit neural representations (e.g., NeRF), it maintains local continuity and differentiability. However, existing works have not fully leveraged these advantageous properties for pose estimation. Instead, current approaches simply treat Gaussian splats as discrete points: Jiang et al. [35] and Sun et al. [36] apply traditional Perspective-n-Point (PnP) methods, while Herau et al. [20] use a hash-encoded MLP [37] to generate 3DGS properties. While these methods have shown promising results, they primarily treat Gaussian splats as point features, leaving the potential benefits of their local continuity yet to be explored.

The proposed method leverages 2DGS because the precision and efficiency of Gaussian splatting can significantly improve the accuracy and speed of the calibration process. Additionally, we demonstrate that the continuity of Gaussian splatting can be effectively used by introducing geometric constraints, which is detailed in Sec. V.

III. OVERVIEW AND PRELIMINARY

Our objective was to estimate LiDAR-camera extrinsic parameters throughout the 3D reconstruction process. In this section, we first outline the problem definition and provide an overview of the proposed method. Then, we introduce 2DGS as preliminary knowledge.

A. Problem Definition and Overview

As illustrated in Fig. 2 , we use multiple sequential LiDAR frames $\{\mathcal{F}_1^{\mathrm{L}}, \mathcal{F}_2^{\mathrm{L}}, \ldots, \mathcal{F}_N^{\mathrm{L}}\}$ with known corresponding poses $\{\mathcal{P}_1^{\mathrm{L}}, \mathcal{P}_2^{\mathrm{L}}, \ldots, \mathcal{P}_N^{\mathrm{L}}\}$ and camera frames $\{\mathcal{F}_1^{\mathrm{C}}, \mathcal{F}_2^{\mathrm{C}}, \ldots, \mathcal{F}_N^{\mathrm{C}}\}$ captured simultaneously as the input, where N is the number of frames. The LiDAR and camera are mounted on a rigid joint, which ensures that the extrinsic parameters between them remain consistent across all sequences. We represent the LiDAR-to-camera pose matrix as $\mathbf{T}^e \in \mathbb{R}^{4\times4}$, defined as follows:

$$\mathbf{T}^e = \begin{pmatrix} \mathbf{R}^e & \mathbf{t}^e \\ \mathbf{0}_{1\times3} & 1 \end{pmatrix}, \tag{1}$$

where $\mathbf{R}^e \in \mathbb{R}^{3\times3}$ is the extrinsic rotation matrix and $\mathbf{t}^e \in \mathbb{R}^3$ is the extrinsic translation vector. An initial rough estimate of $\mathbf{T}^e$ is provided as input, with rotation parameters that allow partial overlap between the LiDAR point cloud projections and the camera's field of view. It will then be refined by our program. Additionally, we assumed that the scene does not contain dynamic objects because all frames were used to reconstruct a static 2DGS.

In the proposed approach, we begin by using LiDAR frames and their corresponding LiDAR poses to train 2DGS, and we optimize only the geometric parameters. After optimizing these geometric parameters, we fix them and then use the camera frames to update the colors of the splats. Throughout this process, we simultaneously estimate the extrinsic parameters between LiDAR and the camera. Fig. 2 illustrates the workflow of the proposed method.

B. Preliminary: 2DGS

2DGS [21] represents a scene as a collection of 2D Gaussian splats with the following learnable parameters:

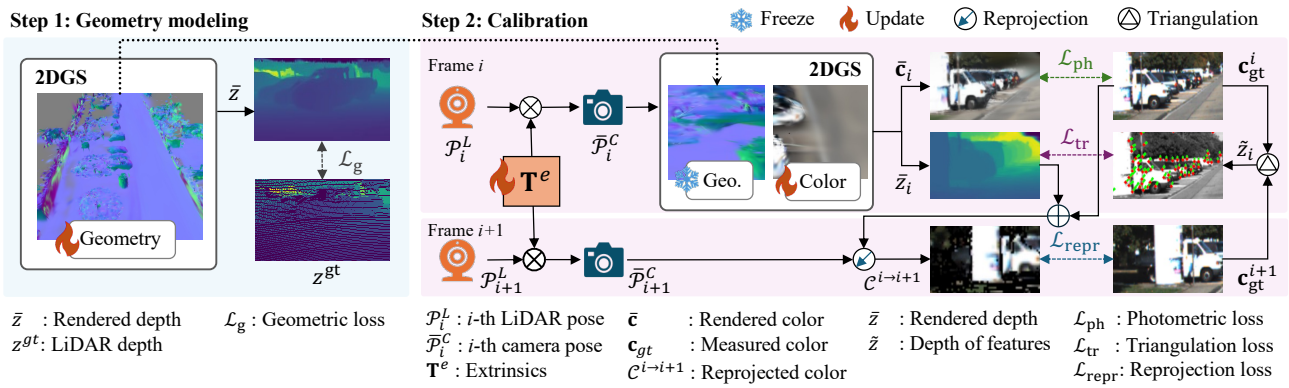


Fig. 2. Workflow of the method. The right panel illustrates how the LiDAR frames were used to supervise the geometric properties of the 2DGS. As shown in the left panel, we freeze the geometric properties and update only the color properties during the calibration process. In addition to photometric loss, we also employ two interframe losses: triangulation loss and reprojection loss.

center position $\mathbf{p}_k \in \mathbb{R}^3 (1 < k \leq N_k)$, opacity $o_k \in [0, 1]$, principal tangential unit vectors $\mathbf{l}_k^u \in \mathbb{R}^3$, $\mathbf{l}_k^v \in \mathbb{R}^3$, and scale factors $s_k^u \in \mathbb{R}$, $s_k^v \in \mathbb{R}$, where N_k is the total number of splats. We define a 2D Gaussian in a world space with the 2D coordinates $\mathbf{u} = (u, v)$ on the tangential plane as follows:

$$P(\mathbf{u}) = \mathbf{p}_k + s_k^u \mathbf{l}_k^u u + s_k^v \mathbf{l}_k^v v. \tag{2}$$

Each splat also has an RGB color $\mathbf{c}_k \in \mathbb{R}^3$. The alpha value at the point $\mathbf{u}$ is given by the Gaussian function $\mathcal{G}_k(\mathbf{u})$ and the opacity of the splat as follows:

$$\alpha_k(\mathbf{u}) = o_k \mathcal{G}_k(\mathbf{u}). \tag{3}$$

To render a 2D splat at a camera pose $\mathcal{P}^{\mathrm{C}} \in \mathbb{R}^{4\times4}$ with an intrinsic projection matrix $\mathbf{K}_{3\times3}$ expanded to a 4×4 matrix as $\mathcal{K} = \begin{pmatrix} \mathbf{K}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{pmatrix}$, the 2D Gaussian P is projected into the camera screen space as follows:

$$P^{\mathrm{C}}(\mathbf{u}) = \mathcal{K}\mathcal{P}^{\mathrm{C}} P(\mathbf{u}) = (xz, yz, z, 1)^{\mathsf{T}}. \tag{4}$$

Here, $\mathbf{x} = (x, y)$ represents the 2D image coordinate, and z is the depth in the camera space. Please refer to the original paper for the inverse-calculation $\mathbf{u}$ from $\mathbf{x}$: $\mathbf{u}(\mathbf{x})$. Finally, the color $\mathbf{c}$ at pixel $\mathbf{x}$ is rendered by alpha blending as follows:

$$\mathbf{c}(\mathbf{x}) = \sum_{k=1}^{N_k^*} \omega_k \mathbf{c}_k, \tag{5}$$

$$\omega_k = \alpha_k(\mathbf{u}(\mathbf{x})) \prod_{j=1}^{k-1} (1 - \alpha_j(\mathbf{u}(\mathbf{x}))). \tag{6}$$

N_k^* denotes the total number of 2D splats intersected with the ray of $\mathbf{x}$. 2DGS optimizes the above learnable parameters using the loss between the rendered color and the color from the input image.

IV. Geometric 2DGS from LiDAR Frames

Unlike SfM with cameras, LiDAR provides absolute scale and precise depth measurements. Thus, we first optimize the geometric properties of 2DGS $(\mathbf{p}_k, \mathbf{l}_k^u, \mathbf{l}_k^v, s_k^u, s_k^v, o_k, \epsilon_k)_{k=1}^{N_k}$ using only the LiDAR frames $\{\mathcal{F}^L\}$. ϵ is an additional parameter representing the depth uncertainty, which is used in the photometric loss (Sec. V). We assume that the corresponding LiDAR poses $\{\mathcal{P}^L\}$ are known, as there are many off-the-shelf point cloud registration methods available [38], [39]. We use the downsampled LiDAR points as the seeds of the Gaussian splats (see details in Sec. VI-A) and introduce a depth-specific splats adaptation method to enhance the quality of surface representation and reduce number of unnecessary.

A. Depth-supervised 2DGS

We optimize the geometric parameters of the 2DGS by using the geometric loss $\mathcal{L}_{\mathrm{g}}$, which is composed of the depth loss $\mathcal{L}_{\mathrm{d}}$, depth uncertainty loss $\mathcal{L}_{\mathrm{unc}}$, depth distortion loss $\mathcal{L}_{\mathrm{dist}}$, and normal consistency loss $\mathcal{L}_{\mathrm{norm}}$:

$$(\mathbf{p}_k, \mathbf{l}_k, s_k, o_k, \epsilon_k)_{k=1}^{N_k} = \underset{(\mathbf{p},\mathbf{l},s,o,\epsilon)}{\arg\min} (\mathcal{L}_{\mathrm{g}}), \tag{7}$$

$$\mathcal{L}_{\mathrm{g}} = \mathcal{L}_{\mathrm{d}} + \mathcal{L}_{\mathrm{unc}} + \lambda_{\mathrm{d}} \mathcal{L}_{\mathrm{dist}} + \lambda_{\mathrm{n}} \mathcal{L}_{\mathrm{norm}}, \tag{8}$$

where λ_{d} and λ_{n} are weight paramters. The definitions of $\mathcal{L}_{\mathrm{dist}}$ and $\mathcal{L}_{\mathrm{norm}}$ follow [21].

1) Depth loss: The depth loss is the difference between the LiDAR depth and the rendered depth from 2DGS. The rays were sampled at all LiDAR points. The ray is cast from the LiDAR position, and its depth is represented as z_i^{L}. The depth loss is defined using the L1 loss between the LiDAR depth and the rendered depth $\bar{z}_i$:

$$\mathcal{L}_{\mathrm{d}} = \frac{1}{N_r} \sum_{i}^{N_r} |\bar{z}_i - z_i^{\mathrm{L}}|, \tag{9}$$

where N_r is the total number of rays. The rendered depth $\bar{z}$ for a LiDAR ray is calculated as the weighted sum of all intersected Gaussian splat depths [21] as follows:

$$\bar{z} = \sum_{k=1}^{N_k^*} \omega_k z_k / \left(\sum_k \omega_k \right), \tag{10}$$

where z_k is the ray-splat intersection depth calculated by Eq. (4), and ω_k is the weight of each splat calculated by Eq. (6).

2) Depth uncertainty loss: To evaluate the errors caused by incorrectly defining the geometry of the 2DGS, we introduce depth uncertainty. The depth error for each ray can be rendered by alpha blending as follows:

$$\bar{e_i} = \sum_{k=1}^{N_k{}^*} \omega_k \epsilon_k. \tag{11}$$

$(\epsilon_k)_{k=1}^{N_k}$ is updated based on the depth error of each ray, which is calculated as $e_i = |\bar{z}_i - z_i^{\mathrm{L}}|$. The uncertainty loss is described using the following L1 loss function:

$$\mathcal{L}_{\text{unc}} = \frac{1}{N_r} \sum_{i}^{N_r} |\bar{e_i} - e_i|. \tag{12}$$

Note that the gradient of $\mathcal{L}_{\text{unc}}$ will not be passed to α_k. Fig. 3 shows an example of the rendered depth error map. The depth errors are relatively large along the depth edges, which is consistent with our expectations.

B. Splats Adaptation

The original 3DGS and 2DGS splats were split and cloned based on the cumulative gradients (errors) of the existing splats. This approach generates new splats only around existing splats and does not guarantee correct propagation of the isolated surfaces.

Thus, the proposed method directly places splats around LiDAR points according to the depth error. If a LiDAR point is detected in a close range that differs significantly from the rendered depth ($\bar{z}_i - z_i^{\mathrm{L}} > \theta_1 (> 0)$), we infer that there are insufficient splats near this LiDAR point; thus, we add a splat at that location. The initial surface normal of these splats aligns with the view direction. Conversely, if the LiDAR point is located behind the rendered depth, the existing foreground splats will adjust accordingly; thus, no further action is required.

V. LiDAR-Camera Calibration with 2DGS

After optimizing the geometric 2DGS, the colors $\mathbf{c}_k$ are estimated from the camera images. Because the camera poses are unknown, we jointly optimize the camera poses $\left(\bar{\mathcal{P}}_i^C\right)_{i=1}^N$ with the colors. However, we only need to find the relative pose $\mathbf{T}^e$ between the camera and LiDAR because they are fixed. Thus, the poses of the camera frames $\left(\bar{\mathcal{P}}_i^C\right)_{i=1}^N$ can be estimated with: $\bar{\mathcal{P}}_i^C = \mathbf{T}^e \cdot \mathcal{P}_i^L$.

A. Color and Pose Joint Optimization

The joint optimization of $\mathbf{c}_k$ and $\mathbf{T}^e$ seeks to minimize the correspondence loss, $\mathcal{L}_{\text{c}}$, which comprises the photometric loss $\mathcal{L}_{\text{ph}}$, triangulation loss $\mathcal{L}_{\text{tr}}$, and reprojection loss $\mathcal{L}_{\text{repr}}$:

$$\mathbf{T}^e, (\mathbf{c}_k)_{k=1}^{N_k} = \underset{\mathbf{T}^e, \{\mathbf{c}_k\}}{\operatorname{argmin}} \mathcal{L}_{\text{c}}, \tag{13}$$

$$\mathcal{L}_{\text{c}} = \mathcal{L}_{\text{ph}} + \lambda_t \mathcal{L}_{\text{tr}} + \lambda_r \mathcal{L}_{\text{repr}}. \tag{14}$$

Fig. 2 illustrates the optimization process. The photometric loss is defined as the L2 distance between the measured pixel color $\mathbf{c}_i^{\mathrm{L}}$ and the rendered color $\bar{\mathbf{c}}_i$ according to Eq. (6). However, the geometric error influences the optimization of the relative pose $\mathbf{T}^e$; thus, we incorporate depth uncertainty weights as follows:

Fig. 3. Example of the rendered depth error. A lighter color indicates a larger value. The depth error map emphasizes the edge components.

$$\mathcal{L}_{\text{ph}} = \frac{1}{\sum w_i} \sum_{i}^{N_r} w_i \left(\bar{\mathbf{c}}_i - \mathbf{c}^{\mathrm{L}}\right)^2, \tag{15}$$

$$w_i = \exp(-\bar{e}_i). \tag{16}$$

As described in [19] and [29], the photometric loss optimizes $\mathbf{T}^e$ while updating $\{\mathbf{c}_k\}$. However, the photometric loss alone does not provide a sufficient update direction to effectively optimize the relative pose. Therefore, in the following sections, we closely examine the limitations of the photometric loss and propose solutions by incorporating the triangulation loss $\mathcal{L}_{\text{tr}}$ and the reprojection loss $\mathcal{L}_{\text{repr}}$.

B. Limitation of Photometric Loss

Because we use the first-order gradient descent method for optimization, we can examine the update directions to $\mathbf{T}^e$ in each iteration by inspecting $\frac{\partial \mathcal{L}_{\text{ph}}}{\partial \mathbf{T}^e}$. We denote the matrix dot product using the Frobenius inner product $\langle \cdot, \cdot \rangle$ [40]. Refer to Appendix for the detailed derivation. By applying the chain rule of the derivatives, we can derive the following equation:

$$\frac{\partial \mathcal{L}_{\text{ph}}}{\partial \mathbf{T}^e_{[mn]}} = \sum_{i}^{N_k^*} \left\langle \frac{\partial \mathcal{L}_{\text{ph}}}{\partial u_i}, \frac{\partial u_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle + \sum_{i}^{N_k^*} \left\langle \frac{\partial \mathcal{L}_{\text{ph}}}{\partial v_i}, \frac{\partial v_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle, \tag{17}$$

where $(u_i,\, v_i)$ represents the intersected point with i-th 2D splat, as in Eq. (4). The subscript $(\cdot)_{[mn]}$ denotes the element at the m-th row and the n-th column. We omit the factor $w_i / \sum w_i$ in the above equation because it is trivial for the analysis.

1) Robustness against initial pose error: Because $\frac{\partial \mathcal{L}_{\text{ph}}}{\partial u_i}$ and $\frac{\partial \mathcal{L}_{\text{ph}}}{\partial v_i}$ are scalars, the spatial update directions are given by $\frac{\partial u_i}{\partial \mathbf{T}^e}$ and $\frac{\partial v_i}{\partial \mathbf{T}^e}$. Applying the chain rule, we derive the following equation:

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{ph}}}{\partial \mathbf{T}^e_{[mn]}} &= \sum_{i}^{N_k{}^*} \left\langle \mathbf{h}_i, \frac{\partial \hat{\mathbf{p}}_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle, \\ \mathbf{h}_i &= \frac{\partial \mathcal{L}_{\text{ph}}}{\partial u_i} \cdot \frac{\mathbf{l}_{u_i}}{s_{u_i}} + \frac{\partial \mathcal{L}_{\text{ph}}}{\partial v_i} \cdot \frac{\mathbf{l}_{v_i}}{s_{v_i}}, \end{aligned} \tag{18}$$

where $\hat{\mathbf{p}}_i$ represents the intersection of the 2D splat with the camera ray, relating to u_i and v_i as per Eq. (2). This implies that each intersection point's update direction, denoted by $\mathbf{h}_i$, is restricted to its respective splat plane. A key issue with this approach is that the gradients are limited to the local region within each splat's size. If the error in $\mathbf{T}^{\mathrm{e}}$ is large

TABLE I

CALIBRATION RESULTS OF THE PROPOSED METHOD AND COMPARISION METHODS. TRANSLATION ERROR* IS CALCULATED WITH SUCCESS CASES.

Method	Near			Far		
	Success Rate %	Rotation Error (°)	Translation Error* (cm)	Success Rate %	Rotation Error (°)	Translation Error* (cm)
Edge [9]	0	127.96 ± 34.21	—	0	119.14 ± 33.50	—
Intensity [11]	63	1.07 ± 1.11	14.2 ± 9.1	10	21.10 ± 32.42	61.2 ± 65.9
Motion [14]	0	129.79 ± 62.42	—	0	131.18 ± 69.93	—
INF [19]	53	5.74 ± 16.33	15.3 ± 8.7	23	6.17 ± 5.92	20.0 ± 17.1
MOISST* [29]	10	3.37 ± 2.37	31.1 ± 14.0	0	11.78 ± 6.12	—
Ours w. Colmap	97	0.39 ± 0.28	9.9 ± 3.1	97	0.43 ± 0.45	10.2 ± 4.3
Ours w. SS	**100**	**0.36 ± 0.15**	**8.7 ± 3.2**	**100**	**0.39 ± 0.18**	**8.8 ± 3.4**

and the camera ray displacement significantly deviates, the update directions of the Gaussian splats may lose relevance.

We address the instability caused by local gradients by applying a reprojection loss (Sec. V-C), which directly retrieves the update direction from 2D image pixels rather than from 3D local spaces.

2) Instability in translation estimation: Let us take a closer look at the extrinsic translation vector $\mathbf{t}^{\mathrm{e}}$. Here, we denote $\mathbf{n}_i$ as the normal vector of i-th Gaussian splat, $\mathbf{R}^{\mathrm{C}}$ as the rotation matrix of the estimated camera pose, and $\mathbf{r}$ as the ray direction in the camera space. We obtain the following equation:

$$\frac{\partial \mathcal{L}_{\text{ph}}}{\partial \mathbf{t}^{e}_{[m]}} = \sum_{i}^{{N_k}^*} \mathbf{h}_{i[m]} - \frac{\langle \mathbf{R}^{C}\mathbf{r}, \mathbf{h}_i \rangle}{\langle \mathbf{R}^{C}\mathbf{r}, \mathbf{n}_i \rangle} \mathbf{n}_{i[m]}, \tag{19}$$

where $(\cdot)_{[m]}$ is the m-th element in the vector.

We find that the update direction of the camera pose includes components that are parallel and perpendicular to the splat surface. However, the coefficient of the normal vector direction is affected by the camera ray direction. As described in [41], surfaces nearly parallel to camera rays are unreliable. Thus, valid Gaussian splats should ideally be angled relative to the view direction, minimizing the perpendicular component's coefficient in Eq. 19. Nonetheless, as discussed in [15], the normal vector direction is critical for photometric loss, especially when there is minimal rotation in LiDAR-camera motion, such as in autonomous driving. In this case, the photometric loss alone is insufficient for accurate translation estimation.

Thus, we propose the use of the triangulation loss, which is described in detail in Sec. V-D.

C. Reprojection Loss

We incorporate the reprojection loss $\mathcal{L}_{\text{repr}}$ to enhance the robustness of the proposed method, particularly for optimizing the rotation component. Although errors in the ray direction can cause significant disparities in the intersected Gaussian splats, the reprojected pixels exhibit minimal displacement in the camera space, which facilitate accurate gradient computation. The displacement decreases further with increasing distance.

For a pixel $\mathbf{v}_i \in \mathbb{R}^2$ in the n-th camera image, we reproject it to the $(n+1)$-th camera image to obtain the pixel coordinate $\mathbf{w}_i \in \mathbb{R}^2$ using the following equation:

$$\begin{pmatrix} \mathbf{w}_i \\ 1 \end{pmatrix} z_i^w = \bar{z}_i \mathbf{R}^{n \to n+1} \mathbf{K}^{-1} \begin{pmatrix} \mathbf{v}_i \\ 1 \end{pmatrix} + \mathbf{t}^{n \to n+1}, \tag{20}$$

where $\mathbf{R}^{n \to n+1}$ and $\mathbf{t}^{n \to n+1}$ represent the rotation matrix and the translation vector from the n-th to the $(n+1)$-th camera space; $\bar{z}_i$ is the rendered depth value of pixel $\mathbf{v}_i$ from its original camera viewpoint, whereas z_i^w is the reprojected depth. We then use a bilinear interpolation function $\mathcal{C}(\cdot)$ to approximate the color of $\mathbf{w}_i$ based on the four nearest pixels. To ensure robustness, we initially downsample the image pixels and progressively reduce the downsampling rate during training. The reprojection loss is defined as follows:

$$\mathcal{L}_{\text{repr}} = \frac{1}{N_r} \sum_{i}^{N_r} \left(\mathcal{C}\left(\mathbf{w}_i\right) - \mathcal{C}\left(\mathbf{v}_i\right) \right)^2. \tag{21}$$

To handle occlusion when projecting from one camera view to another, we first render the depth maps from viewpoints $\mathbf{T}_1^C$ and $\mathbf{T}_2^C$. Then, we reproject the rendered depth from one view to another using Eq. (20) and compare z_i^w and the rendered depth $\bar{z}_i^w$ from the corresponding viewpoint. If the difference between these values exceeds a threshold value θ_2, such points are excluded from the reprojection loss calculation.

D. Triangulation Loss

We propose applying the triangulation loss to the calibration with 2DGS. We assume that we can retrieve M corresponding feature points in a sequential frames with coordinates $(\mathbf{q}_n^i)_{i=1}^M$ and $(\mathbf{q}_{n+1}^i)_{i=1}^M$, where each $\mathbf{q}_n^i, \mathbf{q}_{n+1}^i \in \mathbb{R}^2$ is from the n-th camera image and the $(n+1)$-th camera image, respectively. For simplicity, we omit the superscript i in the following equations. Given $\mathbf{q}_n$ and $\mathbf{q}_{n+1}$, we can find the intersection point $\mathbf{Q} \in \mathbb{R}^3$ at which the rays from their respective camera positions in the world space intersect by

$$\mathbf{Q} = \mathbf{T}_n^C \mathbf{K}^{-1} \begin{pmatrix} \mathbf{q}_n \\ 1 \end{pmatrix} \tilde{z}_n = \mathbf{T}_{n+1}^C \mathbf{K}^{-1} \begin{pmatrix} \mathbf{q}_{n+1} \\ 1 \end{pmatrix} \tilde{z}_{n+1}, \tag{22}$$

where $\tilde{z}_n$ and $\tilde{z}_{n+1}$ are the two z-depths in the camera space from the camera position to $\mathbf{Q}$. Denoting the m-th row of a

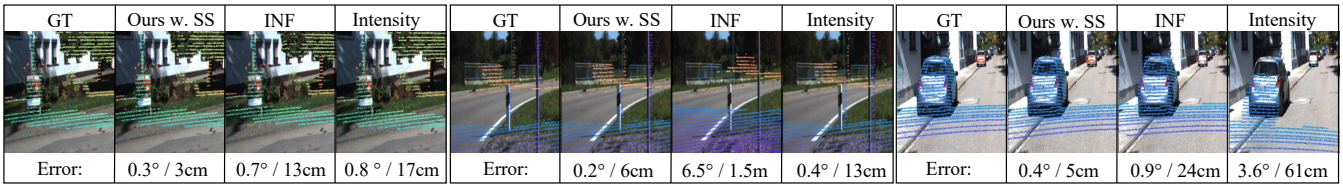


Fig. 4. Ground truth image with LiDAR points projected using the resulting extrinsic parameters. An error of several degrees can cause significant displacement, whereas even an error of 20 cm does not result in noticeable displacement. Among the compared methods, the proposed method demonstrated robust results.

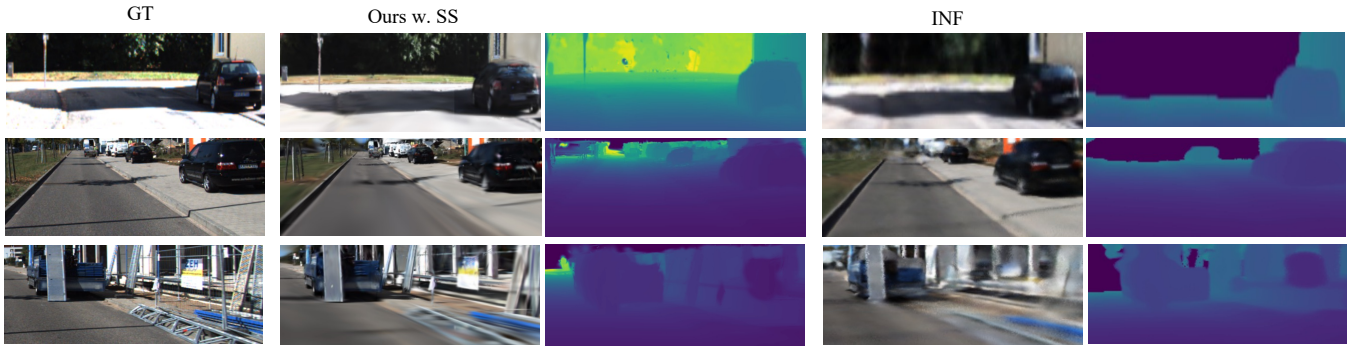


Fig. 5. Rendered RGB and depth maps using proposed method and INF. The rendering quality of 2DGS (Ours) was much higher than that of MLP (INF). Because volume rendering samples points within a certain range along a ray, points in distant areas are not sampled, resulting in those areas being unseen in the INF rendered image.

matrix as $\cdot_{[m]}$, we can formulate as follows:

$$\frac{\mathbf{R}_{[1]}^{n\to n+1}\mathbf{K}^{-1}\begin{pmatrix}\mathbf{q}_n\\1\end{pmatrix}\tilde{z}_n^x+\mathbf{t}_{[1]}^{n\to n+1}}{\mathbf{R}_{[3]}^{n\to n+1}\mathbf{K}^{-1}\begin{pmatrix}\mathbf{q}_n\\1\end{pmatrix}\tilde{z}_n^x+\mathbf{t}_{[3]}^{n\to n+1}}=(\mathbf{K}^{-1})_{[1]}\begin{pmatrix}\mathbf{q}_{n+1}\\1\end{pmatrix}. \tag{23}$$

We can derive $\tilde{z}_n^x$ from the above equation, which provides an example of how to use the x-coordinates to determine the depth $\tilde{z}_n$. Similarly, we can formulate another equation by using the second row $\cdot_{[2]}$ instead of the first row $\cdot_{[1]}$ of the matrices to use the y-coordinates to calculate the z-depth, denoting as $\tilde{z}_n^y$.

We describe the triangulation loss by comparing the calculated $\tilde{z}_n$ with rendered depth $\bar{z}_i$, as follows:

$$\mathcal{L}_{\text{tr}}=\frac{1}{2}\cdot\frac{1}{N_r}\sum_{i}^{N_r}\left(\mathcal{T}\left(\tilde{z}_n^x,\bar{z}_i\right)+\mathcal{T}\left(\tilde{z}_n^y,\bar{z}_i\right)\right), \tag{24}$$

where $\mathcal{T}(\cdot,\cdot)$ denotes the Tukey robust function, which mitigates the influence of outliers. The threshold of Tukey loss is set to 1 m in this experiment. Because $\tilde{z}_n$ and $\bar{z}_i$ are functions of the extrinsic parameters $\mathbf{T}^e$, the gradients are backpropagated through both quantities.

VI. EXPERIMENTS

A. Dataset and experimental settings

We used 30 sequences in the KITTI odometry dataset [42], each comprising 50 sequential LiDAR-camera pairs without dynamic objects. The extrinsic parameters in $\mathbf{T}^e$ are expressed using the Lie algebra in $\mathfrak{se}3$. To simulate different levels of the initial calibration error, we introduced a "far" initial bias by adding 0.2 to all the $\mathfrak{se}3$ parameters, resulting in an error of $16.84°/29.25cm$. The scalar rotation error is defined as the magnitude of the error represented by the rotation vector. Additionally, we applied a "near" initial bias by adding 0.1 to the translation parameters while leaving the rotation parameters unchanged, leading to an error of $0°/14.68cm$. On an RTX4090 GPU, our method required 2.0 and 6.8 minutes for geometry optimization and calibration, respectively. For more results on other dataset, please refer to the Appendix.

In the process of generating geometric Gaussian splats from the LiDAR frames, we begin by separating the ground points from nonground points. The ground points were downsampled using a 0.5 m voxel grid, while the remaining points were downsampled using a finer 0.15 m voxel grid. These downsampled points serve as the initial Gaussian splats. We densify the points by cloning them four times in the final RGB rendering. We set the threshold value θ_1 for splats adaptation to 0.5 m and θ_2 for occlusion handling to 0.05 m. We set weight parameters λ_{d}, λ_{n}, λ_t, and λ_r to $1e^4$, $1e^{-1}$, 1, and 200, respectively.

We employed the Adam optimizer [43] for the optimization process. The learning rates of the Gaussian properties were set in accordance with the original 2DGS [21]. To address the sensitivity of the translation parameters to the initial rotation errors, we initialized the learning rate for the rotation parameters at $1e^{-2}$ and for the translation parameters at $5e^{-4}$. Once the rotation change was less than $0.1°$ over the last 500 iterations, we adjusted the learning rates, setting the rotation to $1e^{-3}$ and the translation to $1e^{-2}$. The learning rate is halved when the change rate of translation falls below $1e^{-5}$

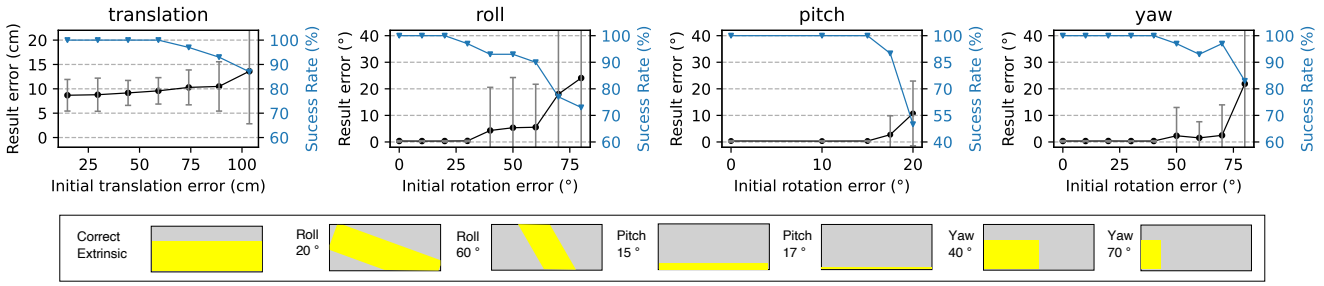


Fig. 6. Success rate and calibration error with respect to different initial biases. Here, we illustrate how the target LiDAR point cloud is reprojected onto the camera image relative to the initial rotation error: the gray background denotes the camera image area, and the yellow foreground denotes the target LiDAR area. The proposed method does not work if the target LiDAR points are initially not visible.

and $5e^{-6}$. We uses 15000 iterations for both the geometric 2DGS generation and joint optimization processes.

B. Evaluation of Calibration Accuracy

We present the results of our calibration method using two feature extraction and matching approaches for triangulation loss $\mathcal{L}_{\text{tr}}$: Colmap features and matching, denoted as "Ours w. Colmap," and SuperPoint [44] with SuperGlue [45], denoted as "Ours w. SS." To evaluate calibration accuracy and robustness, we compared our method with the following approaches:

- Edge [9]: Align the projected LiDAR edges with the camera edges.
- Intensity [11]: Maximize the mutual information between the reflectivity of the LiDAR scans and the intensity of the camera images. In the "near" experiment, we only applied the refinement process described in [11]; in the "far" experiment, we applied the automatic initial guess in [11].
- Motion [14]: Estimate camera motion with LiDAR point cloud and extrinsic parameters using hand-eye calibration.
- INF [19]: Optimize the pose during NeRF [17] reconstruction.
- MOISST* [29]: Optimize the pose during Instant-NGP [37] reconstruction. Implemented ourselves.

Because rotation errors can heavily affect translation, we considered results with rotation errors under 1° as successful and excluded failures from the translation error calculation. * indicates errors calculated only for successful cases. Thus, the translation and rotation errors represent accuracy, whereas the success rate reflects robustness. Table I provides the detailed results, and Fig. 4. shows a qualitative reprojected example.

The results demonstrate that the proposed method outperforms existing approaches. Appearance-based methods face challenges with the domain gap between LiDAR and camera RGB data because LiDAR reflectivity does not always correlate with RGB intensity, and depth or normal edges in LiDAR scans differ from texture edges in RGB images. Additionally, the KITTI dataset is noisy and lacks structure; thus, edge-based methods are difficult to extract meaningful features. Motion-based calibration requires rotational motion in multiple directions, which hinders convergence when the data includes large linear motions. INF [19] exhibits low reconstruction quality (see Fig. 5), while hash encoding methods like [29] struggle with large initial rotation errors. In contrast, the proposed method demonstrate robust performance even with initial errors, and the translation errors remained within acceptable limits.

C. Ablation Study on Losses

In this section, we demonstrate the effectiveness of the triangulation loss, reprojection loss, and depth uncertainty weights. We conducted ablation studies by removing each of these components in the optimization process, starting with the "far" initial extrinsic parameters. Table II lists the results. The rows, from top to bottom, represent the calibration results for: our method without triangulation loss, using Colmap feature points for triangulation loss, without reprojection loss, without depth uncertainty weights, and using SuperPoint-SuperGlue feature points for triangulation loss.

We can notice that the results highlight the critical role of reprojection loss, where its removal leads to a significant drop in robustness, reducing the success rate to 30%. The triangulation loss and depth uncertainty weights improve the accuracy by minimizing the translation errors. Interestingly, the translation error remained largely unchanged whether the Colmap-matched points are used or the triangulation loss is entirely omitted. This outcome aligns with the analysis in Sec. V-B, where the focus is on points on surfaces, whereas the Colmap features are predominantly edge points, which are often filtered out by the Tukey loss due to the instability of the depth in edges. Additionally, we conducted experiments using 3DGS as the base representation, with detailed results presented in the Appendix.

D. Evaluation of the Calibration Robustness

To account for the sensitivity of the initial rotation error to the axes, we separately evaluated the impact of the initial rotation errors along each axis on the calibrated rotation parameters. For rotation errors, we define success cases as those with an error within 1°, as in the previous experiments. For the translation results, errors within 20 cm were considered to be successful. Fig. 6 shows the results. In our experiment,

TABLE II

ABLATION EXPERIMENT RESULTS WITH DIFFERENT EXPERIMENT SETTINGS.

Method	Success Rate %	Rotation Error (°)	Translation Error* (cm)
Ours wo. triangulation	97	0.44±0.38	11.18±3.4
Ours colmap	97	0.44 ± 0.45	10.16 ± 4.34
Ours wo. reprojection	30	9.28±12.06	11.67±3.14
Ours wo. weights	**100**	0.40±0.16	9.93±4.22
Ours full	**100**	**0.39 ± 0.18**	**8.79 ± 3.40**

our method successfully converged to the correct pose from the initial errors of 20° (roll), 15° (pitch), 40° (yaw), and 60 cm in translation. In addition, we achieved a 90% success rate for initial errors within 60° (roll), 17° (pitch), 70° (yaw), and 80 cm in translation.

VII. CONCLUSIONS

This paper presents a LiDAR-camera calibration method that uses 2D Gaussian Splatting (2DGS). The proposed method begins by constructing the 2DGS geometry from the LiDAR scans, thereby creating a geometric scene representation. The calibration is then integrated into the 2DGS colorization stage to align the LiDAR data with the captured camera images. We analyzed the limitations of 2DGS in the pose estimation task and addressed them by introducing triangulation and reprojection losses, along with a depth uncertainty weighting scheme to enhance the calibration stability. We validated the proposed method on the challenging KITTI odometry dataset. The results demonstrated that the proposed method improves alignment accuracy and robustness, outperforming existing methods in terms of precision and handling of complex scenarios.

REFERENCES

[1] J. H. Yoo, Y. Kim, J. Kim, and J. W. Choi, "3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection," in *ECCV*. Springer, 2020, pp. 720–736.

[2] H. Zhong, H. Wang, Z. Wu, C. Zhang, Y. Zheng, and T. Tang, "A survey of lidar and camera fusion enhancement," *Procedia Computer Science*, vol. 183, pp. 579–588, 2021.

[3] A. Carlson, M. S. Ramanagopal, N. Tseng, M. Johnson-Roberson, R. Vasudevan, and K. A. Skinner, "Cloner: Camera-lidar fusion for occupancy grid-aided neural representations," *Ra-L*, vol. 8, no. 5, pp. 2812–2819, 2023.

[4] S. Verma, J. S. Berrio, S. Worrall, and E. Nebot, "Automatic extrinsic calibration between a camera and a 3d lidar using 3d point and plane correspondences," in *2019 ITSC*. IEEE, 2019, pp. 3906–3912.

[5] Y. Zhao, K. Huang, H. Lu, and J. Xiao, "Extrinsic calibration of a small fov lidar and a camera," in *2020 CAC*. IEEE, 2020, pp. 3915–3920.

[6] S. Sim, J. Sock, and K. Kwak, "Indirect correspondence-based robust extrinsic calibration of lidar and camera," *Sensors*, vol. 16, no. 6, p. 933, 2016.

[7] H. Yang, X. Liu, and I. Patras, "A simple and effective extrinsic calibration method of a camera and a single line scanning lidar," in *ICPR2012*. IEEE, 2012, pp. 1439–1442.

[8] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers." in *Robotics: science and systems*, vol. 2, no. 7. Berlin, Germany, 2013.

[9] C. Yuan, X. Liu, X. Hong, and F. Zhang, "Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments," *RA-L*, vol. 6, no. 4, pp. 7517–7524, 2021.

[10] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information," in *AAAI*, 2012.

[11] K. Koide, S. Oishi, M. Yokozuka, and A. Banno, "General, single-shot, target-less, and automatic lidar-camera extrinsic calibration toolbox," *ICRA*, 2023.

[12] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks," in *IROS*. IEEE, 2018, pp. 1110–1117.

[13] X. Lv, B. Wang, Z. Dou, D. Ye, and S. Wang, "Lccnet: Lidar and camera self-calibration using cost volume network," in *CVPR*, 2021, pp. 2894–2901.

[14] R. Ishikawa, T. Oishi, and K. Ikeuchi, "Lidar and camera calibration using motions estimated by sensor fusion odometry," in *IROS*. IEEE, 2018, pp. 7342–7349.

[15] R. Ishikawa, S. Zhou, Y. Sato, T. Oishi, and K. Ikeuchi, "Lidar-camera calibration using intensity variance cost," in *ICRA*, 2024.

[16] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *ECCV*, 2016.

[17] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[18] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM TOG*, vol. 42, no. 4, pp. 1–14, 2023.

[19] S. Zhou, S. Xie, R. Ishikawa, K. Sakurada, M. Onishi, and T. Oishi, "Inf: Implicit neural fusion for lidar and camera," in *IROS*. IEEE, 2023, pp. 10 918–10 925.

[20] Q. Herau, M. Bennehar, A. Moreau, N. Piasco, L. Roldao, D. Tsishkou, C. Migniot, P. Vasseur, and C. Demonceaux, "3dgs-calib: 3d gaussian splatting for multimodal spatiotemporal calibration," *arXiv preprint arXiv:2403.11577*, 2024.

[21] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH 2024 Conference Papers*, 2024, pp. 1–11.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[23] J. Kang and N. L. Doh, "Automatic targetless camera–lidar calibration by aligning edge with gaussian mixture model," *Journal of Field Robotics*, vol. 37, no. 1, pp. 158–179, 2020.

[24] X. Zhang, S. Zhu, S. Guo, J. Li, and H. Liu, "Line-based automatic extrinsic calibration of lidar and camera," in *ICRA*. IEEE, 2021, pp. 9347–9353.

[25] J. Shi, Z. Zhu, J. Zhang, R. Liu, Z. Wang, S. Chen, and H. Liu, "Calibrcnn: Calibrating camera and lidar by recurrent convolutional neural network and geometric constraints," in *2020 IROS*. IEEE, 2020, pp. 10 197–10 202.

[26] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "Regnet: Multimodal sensor registration using deep neural networks," in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 1803–1810.

[27] Z. Taylor and J. Nieto, "Motion-based calibration of multimodal sensor extrinsics and timing offset estimation," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1215–1229, 2016.

[28] H.-J. Chien, R. Klette, N. Schneider, and U. Franke, "Visual odometry driven online calibration for monocular lidar-camera systems," in *2016 ICPR*. IEEE, 2016, pp. 2848–2853.

[29] Q. Herau, N. Piasco, M. Bennehar, L. Roldão, D. Tsishkou, C. Migniot, P. Vasseur, and C. Demonceaux, "Moisst: Multimodal optimization of implicit scene for spatiotemporal calibration," in *IROS*. IEEE, 2023, pp. 1810–1817.

[30] S. Ullman, "The interpretation of structure from motion," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979.

[31] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "Barf: Bundle-adjusting neural radiance fields," in *ICCV*, 2021, pp. 5741–5751.

[32] W. Bian, Z. Wang, K. Li, J.-W. Bian, and V. A. Prisacariu, "Nope-nerf: Optimising neural radiance field with no pose prior," in *CVPR*, 2023, pp. 4160–4169.

[33] S. Xie, S. Zhou, K. Sakurada, R. Ishikawa, M. Onishi, and T. Oishi, "G2fr: Frequency regularization in grid-based feature encoding neural radiance fields," *ECCV*, 2024.

[34] L. Xiangjie, X. Shuxiang, K. Sakurada, S. Ryusuke, and O. Takeshi, "Implicit neural fusion of rgb and far-infrared 3d imagery for invisible scenes," *IROS*, 2024.

[35] P. Jiang, G. Pandey, and S. Saripalli, "3dgs-reloc: 3d gaussian splatting for map representation and visual relocalization," *arXiv preprint arXiv:2403.11367*, 2024.
[36] L. C. Sun, N. P. Bhatt, J. C. Liu, Z. Fan, Z. Wang, T. E. Humphreys, and U. Topcu, "Mm3dgs slam: Multi-modal 3d gaussian splatting for slam using vision, depth, and inertial measurements," *arXiv preprint arXiv:2404.00923*, 2024.
[37] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM TOG*, vol. 41, no. 4, pp. 1–15, 2022.
[38] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *ICCV*, 2019, pp. 3523–3532.
[39] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 314–333, 2020.
[40] M. Giles, "An extended collection of matrix derivative results for forward and reverse mode automatic differentiation," 2008.
[41] J. Zhang, S. Singh, *et al.*, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.
[42] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *2012 ICRA*. IEEE, 2012, pp. 3936–3943.
[43] D. Kingma, "Adam: a method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[44] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *CVPR*, 2018, pp. 224–236.
[45] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *CVPR*, 2020, pp. 4938–4947.

APPENDIX

A. Derivation of Gradients

Following the notation in Section V-B, we use Frobenius inner product $< \cdot, \cdot >$ to denote matrix dot product. Following Eq. (17), the update direction for $\mathbf{T}^e$ is determined by weighted sum of $\frac{\partial u_i}{\partial \mathbf{T}^e}$ and $\frac{\partial v_i}{\partial \mathbf{T}^e}$.

In the following equations, we use the subscript $(\cdot)_{[mn]}$ to denote the element at m-th row and the n-th column. Similarly, the subscript $(\cdot)_{[m]}$ denotes the element at m-th row for one-column vector or the element at m-th column for one-row vector. Since $\frac{\partial u_i}{\partial \mathbf{T}^e}$ and $\frac{\partial v_i}{\partial \mathbf{T}^e}$ are symmetric, we will only give a detailed look into $\frac{\partial u_i}{\partial \mathbf{T}^e}$.

$$\begin{aligned}\frac{\partial u_i}{\partial \mathbf{T}^e_{[mn]}} &= \left\langle \frac{\partial u_i}{\partial \hat{\mathbf{p}}_i}, \frac{\partial \hat{\mathbf{p}}_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle \\ &= \left\langle \frac{\mathbf{l}_{u_i}}{s_{u_i}}, \frac{\partial \hat{\mathbf{p}}_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle.\end{aligned} \tag{25}$$

$\hat{\mathbf{p}}_i$ means the intersection point in world space for the 2D splat the the camera ray and u_i is can be calculated by

$$u_i = (\hat{\mathbf{p}}_i - \mathbf{p}_i) \cdot \frac{\mathbf{l}_{u_i}}{s_{u_i}}. \tag{26}$$

Frobenius inner product has property that

$$a \langle \mathbf{A}, \mathbf{B} \rangle = \langle a\mathbf{A}, \mathbf{B} \rangle \tag{27}$$

for a real number a. And $\frac{\partial \mathcal{L}_{\text{ph}}}{\partial u_i}$ is a real scalar number. Thus, we can write

$$\left\langle \frac{\partial \mathcal{L}_{\text{ph}}}{\partial u_i}, \frac{\partial u_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle = \left\langle \frac{\partial \mathcal{L}_{\text{ph}}}{\partial u_i} \cdot \frac{\mathbf{l}_{u_i}}{s_{u_i}}, \frac{\partial \hat{\mathbf{p}}_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle. \tag{28}$$

v_i part can be derived in the same way:

$$\left\langle \frac{\partial \mathcal{L}_{\text{ph}}}{\partial v_i}, \frac{\partial v_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle = \left\langle \frac{\partial \mathcal{L}_{\text{ph}}}{\partial v_i} \cdot \frac{\mathbf{l}_{v_i}}{s_{v_i}}, \frac{\partial \hat{\mathbf{p}}_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle. \tag{29}$$

Thanks to the property of Frobenius inner product that

$$\langle \mathbf{A} + \mathbf{B}, \mathbf{C} \rangle = \langle \mathbf{A}, \mathbf{C} \rangle + \langle \mathbf{B}, \mathbf{C} \rangle, \tag{30}$$

we can derive Eq. (18):

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{ph}}}{\partial \mathbf{T}^e_{[mn]}} &= \sum_i^{N_k} \left\langle \mathbf{h}_i, \frac{\partial \hat{\mathbf{p}}_i}{\partial \mathbf{T}^e_{[mn]}} \right\rangle, \\ \text{where } \mathbf{h}_i &= \frac{\partial \mathcal{L}_{\text{ph}}}{\partial u_i} \cdot \frac{\mathbf{l}_{u_i}}{s_{u_i}} + \frac{\partial \mathcal{L}_{\text{ph}}}{\partial v_i} \cdot \frac{\mathbf{l}_{v_i}}{s_{v_i}}\end{aligned}$$

Then, we will only examine the extrinsic translation vector $\mathbf{t}^e_{[m]}$. We can express the intersected point in another way:

$$\hat{\mathbf{p}}_i = \mathbf{t}^C + \mathbf{R}^C \mathbf{r} z, \tag{31}$$

where $\mathbf{t}^C$ denotes translation vector of the camera pose, $\mathbf{R}^C$ denotes the rotation matrix of the camera pose, and $\mathbf{r}$ denotes the ray direction in camera space. Note that only $\mathbf{t}^C$ is related with $\mathbf{t}^e$ by

$$\mathbf{t}^C = \mathbf{t}^e + \mathbf{R}^e \mathbf{t}^L. \tag{32}$$

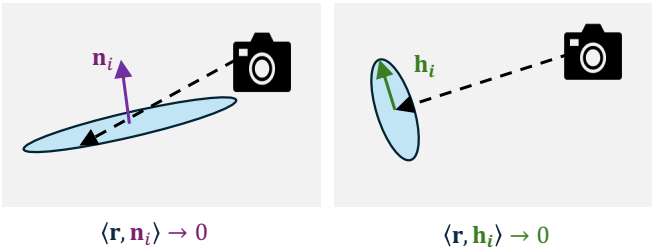


Fig. 7. An illustration of dot product of camera ray with $\mathbf{h}_i$ or $\mathbf{n}_i$.

Thus,

$$\frac{\partial \mathbf{t}^C}{\partial \mathbf{t}^e} = \mathbf{1}. \tag{33}$$

As mentioned in the paper, z can be derived from

$$\begin{aligned}\left\langle \mathbf{t}^C + \mathbf{R}^C \mathbf{r} z - \mathbf{p}_i, \mathbf{n}_i \right\rangle &= 0 \\ \hat{\mathbf{p}}_i = \mathbf{t}^C + \mathbf{R}^C \mathbf{r} \frac{\left\langle \mathbf{p}_i - \mathbf{t}^C, \mathbf{n}_i \right\rangle}{\left\langle \mathbf{R}^C \mathbf{r}, \mathbf{n}_i \right\rangle},\end{aligned} \tag{34}$$

where $\mathbf{n}_i$ and $\mathbf{p}_i$ denote the normal vector and the center position of the i-th Gaussian splat, respectively. Thus, we can obtain

$$\frac{\partial \hat{\mathbf{p}}_i}{\partial \mathbf{t}^C_{[m]}} = \mathbf{1} - \frac{(\mathbf{R}^C \mathbf{r}) \cdot n_{i[m]}}{\left\langle \mathbf{R}^C \mathbf{r}, \mathbf{n}_i \right\rangle}, \tag{35}$$

and by leveraging the property for real matrices:

$$\langle \mathbf{A}, \mathbf{B} \rangle = \langle \mathbf{B}, \mathbf{A} \rangle, \tag{36}$$

we can obtain:

$$\left\langle \mathbf{h}_i, \frac{\partial \hat{\mathbf{p}}_i}{\partial \mathbf{t}^C_{[m]}} \right\rangle = \langle \mathbf{h}_i, \mathbf{1} \rangle - \frac{n_{i[m]}}{\left\langle \mathbf{R}^C \mathbf{r}, \mathbf{n}_i \right\rangle} \cdot \left\langle \mathbf{R}^C \mathbf{r}, \mathbf{h}_i \right\rangle. \tag{37}$$

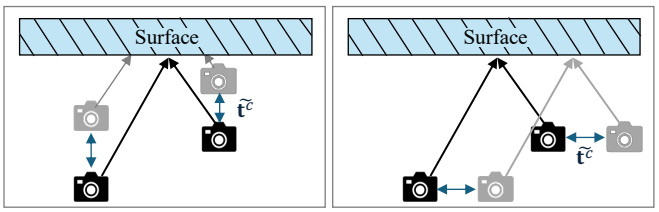
Finally, we can derive Eq. (19):

$$\frac{\partial \mathcal{L}_{\text{ph}}}{\partial \mathbf{t}^C_{[m]}} = \sum_i^{N_k} \mathbf{h}_{i[m]} - \frac{\left\langle \mathbf{R}^C \mathbf{r}, \mathbf{h}_i \right\rangle}{\left\langle \mathbf{R}^C \mathbf{r}, \mathbf{n}_i \right\rangle} \mathbf{n}_{i[m]}. \tag{38}$$

As shown in Fig. 7, Gaussian splats that are perpendicular to the camera ray yield smaller values of $\frac{\left\langle \mathbf{R}^C \mathbf{r}, \mathbf{h}_i \right\rangle}{\left\langle \mathbf{R}^C \mathbf{r}, \mathbf{n}_i \right\rangle}$. In contrast, Gaussian splats that are parallel to the camera rays have larger values. However, the more a Gaussian splat is parallel to the camera ray, the less reliable it is, as even a slight displacement of the camera ray can lead to a completely different intersection point on the splat. Additionally, in the backpropagation implementation, to avoid NaN or infinite gradients, the intersection point values are clipped if they are touching a Gaussian splat that is too parallel to the camera ray, resulting in zero gradients for those points. Therefore, Gaussian splats that are parallel to the camera rays are less reliable and should ideally contribute less to the gradients.

B. Perpendicular Direction for Photometric Loss

In Section V-B, we mentioned that the authors in [15] emphasized the importance of the direction perpendicular to the surface for pose estimation tasks. Here, we will explain how this aligns with our problem settings.



(a) (b)

Fig. 8. An example of the influence of an incorrect extrinsic translation vector on the intersection point is shown in the following scenarios: In (a), the translation error is perpendicular to the surface, causing different intersection points. In (b), the translation error is parallel to the surface, resulting in the intersection points of the two rays still being on the surface.

In our problem setting, we assume that the world-to-LiDAR poses $\mathcal{P}_i^L$ and $\mathcal{P}_{i+1}^L$ are known for the i-th and $(i+1)$-th frames. By using the LiDAR-to-camera extrinsic parameters, the world-to-camera pose is estimated as follows:

$$\bar{\mathcal{P}_i^C} = \mathbf{T}^e \mathcal{P}_i^L = \tilde{\mathbf{T}}^e \mathcal{P}_i^C. \tag{39}$$

We denote the error in the extrinsic parameter as $\tilde{\cdot}$, and $\mathcal{P}_i^C$ represents the ground truth camera pose. Similarly, for the $(i+1)$-th frame, we can write $\bar{\mathcal{P}_{i+1}^C} = \tilde{\mathbf{T}}^e \mathcal{P}_{i+1}^C$.

When the camera poses are correct, a target point $\mathbf{q}$ in 3D space will be projected to the point $\mathbf{K}\mathcal{P}_{i+1}^C\mathbf{q}$ on the $(i+1)$-th camera image and to the point $\mathbf{K}\mathcal{P}_i^C\mathbf{q}$ on the i-th camera image. Since they correspond to the same 3D point, they will have the same colors. In this way, the Gaussian splat containing $\mathbf{q}$ will be updated to the corresponding color.

However, when the camera poses are incorrect, the camera space point $\mathcal{P}_i^C\mathbf{q}$ will lie on a different ray in world space, as shown below:

$$\begin{aligned}
\mathbf{r}_i(x) &= \left(\tilde{\mathbf{R}}^e \mathbf{R}_i^C\right)^{-1} \left(\mathcal{P}_i^C \mathbf{q} x - \tilde{\mathbf{R}}^e \mathbf{t}_i^C - \tilde{\mathbf{t}^e}\right), \\
\mathbf{r}_{i+1}(x) &= \left(\tilde{\mathbf{R}}^e \mathbf{R}_{i+1}^C\right)^{-1} \left(\mathcal{P}_{i+1}^C \mathbf{q} x - \tilde{\mathbf{R}}^e \mathbf{t}_{i+1}^C - \tilde{\mathbf{t}^e}\right)
\end{aligned} \tag{40}$$

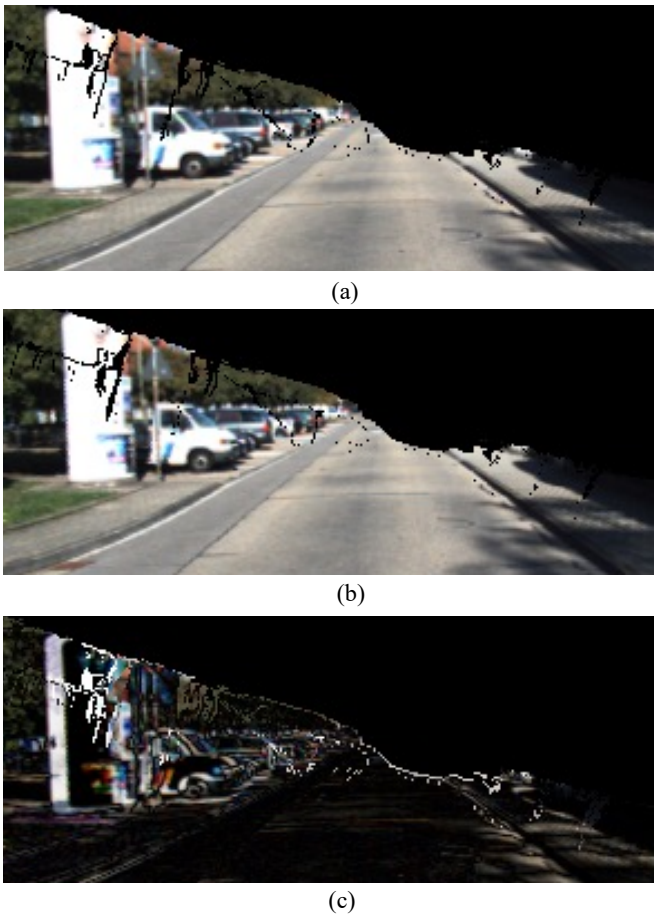
In most cases, these two incorrect rays will not intersect each other on the original Gaussian splat, leading to incorrect color updates for the Gaussian splats. This results in the colors of the Gaussian splats becoming blurred due to the displacement of the camera rays, causing photometric loss. The gradients will then lead the intersection points to move along the Gaussian splats towards more accurate positions. This photometric error will not be reduced unless the camera poses are correct.

However, there is a case when $\mathbf{r}_i(x)$ and $\mathbf{r}_{i+1}(x)$ will intersect a certain Gaussian splat at the same position. Consider the scenario where the rotation error $\tilde{\mathbf{R}}^e$ has already been corrected, leaving only the translation error:

$$\begin{aligned}
\mathbf{r}_i(x) &= \mathbf{q}x + \left(\mathbf{R}_i^C\right)^{-1} \left(\mathbf{t}_i^C (x-1) - \tilde{\mathbf{t}^e}\right), \\
\mathbf{r}_{i+1}(x) &= \mathbf{q}x + \left(\mathbf{R}_{i+1}^C\right)^{-1} \left(\mathbf{t}_{i+1}^C (x-1) - \tilde{\mathbf{t}^e}\right)
\end{aligned} \tag{41}$$

Originally, $\mathbf{r}_i(1) = \mathbf{r}_{i+1}(1)$ if the extrinsic parameters are correct. However, $\mathbf{r}_i(1) = \mathbf{r}_{i+1}(1)$ can still hold true if the following condition is satisfied:

$$\left(\mathbf{R}_i^C\right)^{-1} \tilde{\mathbf{t}^e} = \left(\mathbf{R}_{i+1}^C\right)^{-1} \tilde{\mathbf{t}^e}. \tag{42}$$



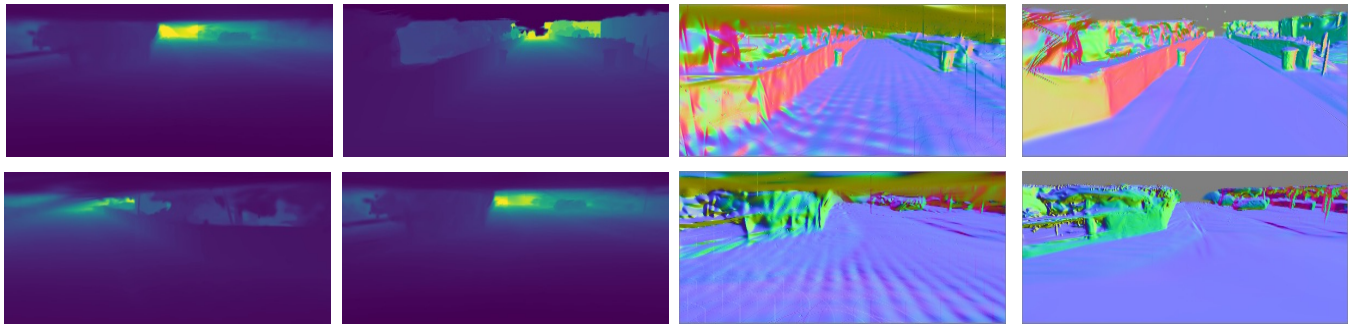
(a) (b) (c)

Fig. 9. Visualization of pixel reprojection under a 16.8° rotation error. (a) Original image points from the first camera view. (b) Points reprojected onto the second camera view. (c) Difference map between (a) and (b), demonstrating larger pixel displacements for nearby points and smaller displacements for distant points due to perspective projection.

This condition always occurs in driving datasets such as KITTI [42], where most echo-poses between inter-frames do not involve rotation. Furthermore, as long as the error term $\tilde{\mathbf{t}^e}$ is parallel to the surface, the intersection point can still lie on the surface, as shown in Fig. 8. In Fig. 8(b), if the translation error is parallel to the surface, the two rays still intersect at the same point on the surface. Consequently, the colors of the intersected Gaussian splats will be updated to the color of $\mathbf{q}$, and there will be no photometric loss due to translation error. However, as mentioned in Eq. (19), during the backpropagation process, few gradients are given when the camera ray direction is perpendicular to the surface, leading to insufficient directions for pose updating.

C. Detail for Reprojection Loss

As discussed in Section V-B, the gradient from photometric loss is constrained within individual splats, which creates inherent limitations in handling large pose errors. This becomes particularly problematic when dealing with large errors in extrinsic parameters $\mathbf{T}^e$, especially in the rotation component $\mathbf{R}^e$, as these errors result in significant



(a) (b) (c) (d)

Fig. 10. (a) Rendered depth image with 3DGS. (b) Rendered depth image with 2DGS. (c) Normal map calculated from (a). (d) Normal map calculated from (b).

TABLE III

CALIBRATION RESULTS USING 3DGS AS BASE REPRESENTATION.

	Success Rate %	Rotation Error (°)	Translation Error* (cm)
Near	30	4.45±3.71	52.71±23.24
Far	10	10.42±10.19	45.26±2.86

displacements for distant points. We estimate the displacement magnitude by $d = 2r\sin(\theta/2)$, where r is the distance to the rotation center and θ is the rotation error. For example, a 10° rotation error causes a displacement of approximately 1.74m for points 10m away. Given Gaussian splats with diameters of about 0.4m, this means the incorrectly intersected splat and the correct splat position may be separated by four or more splats, creating a significant gap in the gradient propagation path.

The photometric loss mechanism operates by updating splat colors based on camera views, following the principle of color consistency across different viewpoints. When the pose estimation is incorrect, intersected splats receive color information from incorrect pixels across different views, leading to erroneous color updates. In the case of a 10° rotation error at 10m distance, this results in color mixing across a wide range of splats, where each splat receives color information from pixels that should correspond to significantly different surface points. This color mixing creates noise in the optimization process, as the gradients computed from these mixed colors may point in arbitrary directions. While having a small portion of splats with unreliable gradients wouldn't significantly impact the overall optimization, the problem becomes critical when the extrinsic error is large. In such cases, a substantial proportion of splats, especially those at greater distances, suffer from this issue. When the majority of splats provide unreliable or misleading gradients, the optimization process may fail to converge to the correct pose.

On the other hand, in the 2D projection space, following the basic principles of perspective projection, distant points occupy smaller areas on the screen. The pixel displacement Δp for a point under rotation error can be approximated

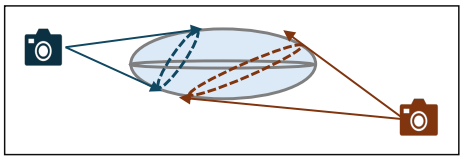


Fig. 11. An illustration of the camera viewed Gaussian splat.

TABLE IV

CALIBRATION ERRORS ON TWO INDOOR SCENES.

	Indoor1 Translation (cm) & Rotation(°) Error			
	x	y	z	r
INF [19]	0.5	**0.6**	2.0	**0.299**
Ours	**0.2**	**0.6**	3.1	0.401
	Indoor2 Translation (cm) & Rotation(°) Error			
	x	y	z	r
INF [19]	**0.4**	1.0	0.9	0.385
Ours	1.5	**0.3**	**0.7**	**0.211**

as $\Delta p \propto f\theta/z$, where f is the focal length, θ is the rotation error, and z is the depth. This relationship shows that despite larger 3D displacements, distant points (larger z) actually result in smaller pixel displacements, as illustrated in Fig. 9. This characteristic of perspective projection makes the reprojection loss particularly effective in compensating for the limitations of 3D space gradients generated by individual splats, as it provides a more controlled gradient signal that naturally scales with depth. The reprojection loss thus serves as a complementary guidance for pose optimization, especially effective for correcting rotation errors affecting distant points where photometric loss struggles.

D. Experiment on 3DGS

We have conducted additional experiments using 3DGS[18] instead of 2DGS.

1) Preliminary: 3DGS use N_k 3D Gaussian splats to represent a scene. Each splat contains: center position $\mathbf{p}_k \in \mathbb{R}^3$, opacity $o_k \in [0, 1]$, scale factors $\mathbf{S}_k \in \mathbb{R}^3$ and a rotation $\mathbf{R}_k$ with quaternion parameterization. 3D covaraince for each

(a) (b) (c) (d)

Fig. 12. Qualitative results of our method. Fisrt row: Indoor1 data; Second row: Indoor 2 data. (a) Rendered depth map with 2DGS after calibration. (b) Rendered color image after calibration. (c) & (d) Reprojected LiDAR points on the images using our calibrated extrinsic parameters and the reference extrinsic parameters.

Gaussian splat is calculated by

$$\mathbf{\Sigma}_k = \mathbf{RSS}^T\mathbf{R}^T. \tag{43}$$

$\mathbf{\Sigma}_k$ is projected to 2D screen space by

$$\mathbf{\Sigma}'_k = \mathbf{JW\Sigma}_k\mathbf{W}^T\mathbf{J}^T, \tag{44}$$

where $\mathbf{W}$ and $\mathbf{J}$ are the view transfomation and the Jacobian matrix that represents the local linear approximation of the projective transformation. Denoting the projected 2D screen space coordinate of $\mathbf{p}_k$ as $\mathbf{x}_k$, the Gaussian value of a pixel $\mathbf{u}$ intersecting k-th Gaussian splat could be calculated with

$$\mathcal{G}_k(\mathbf{u}) = e^{-\frac{1}{2}(\mathbf{u}-\mathbf{x}_k)^T{\mathbf{\Sigma}'_k}^{-1}(\mathbf{u}-\mathbf{x}_k)}. \tag{45}$$

Finally, we can calculate each $\alpha_k(\mathbf{u}) = o_k\mathcal{G}(\mathbf{u})$. The alpha-blending for colors is the same with Eq. 6.

However, we need to notice that the official implementation of 3DGS calculates the expected inverse depth of a pixel using:

$$\frac{1}{\bar{z}} = \sum_{k=1}^{N_k^*}\omega_k\frac{1}{z_k}, \tag{46}$$

where z_k represents the depth of the intersected Gaussian splat's center position. In our experiment, we follow this implementation.

2) Experimental Results: In our implementation, we did not include the depth distortion loss λ_{dist} (because depths are not calculated as the intersected point) and the normal consistency loss λ_{normal} (because normal vector is not defined for 3D sphere).

Our comparative experiments using 3DGS as the base representation (Table III) revealed decreased robustness and accuracy compared to our 2DGS pipeline.

3) Discussion: Intuitively, we can expect this result because a correct surface representation (2DGS) should be more suitable for pose estimation task than an approximate surface representation (3DGS). In detail, we identified two primary limitations.

a) Depth Reconstruction Reliability: The fundamental issue of 3D Gaussian splat is that the intersection of a Gaussian splat and a ray produces a 1-D Gaussian function, making precise depth determination theoretically impossible. What's more, the original 3DGS algorithm's approach of accumulating Gaussian centers' depths leads to imprecise depth calculations. In contrast, 2DGS treats each element as a splat and calculates depth at the intersection point, enabling more accurate depth determination. Furthermore, 3DGS is incompatible with λ_{dist} and λ_{normal}. Without normal regularization, relying solely on depth loss gradients proves insufficient for proper splat orientation. While various works attempt to regularize 3DGS surfaces, 2DGS offers a more theoretically sound approach. The geometry comparison is shown in Fig. 10.

b) View-Dependent Projection Issues: A secondary but crucial limitation is that 2D projections of 3D Gaussian splats don't precisely represent the actual surface. As illustrated in Fig. 11, when a 3D Gaussian splat approximates a surface, its cross-section on the camera view plane varies with camera position and viewing direction. This view-dependency means that the same surface appears different from various view-

points, introducing additional complexity to our calibration pipeline.

E. Experiments on Indoor Dataset

We have conducted additional experiments using the indoor dataset from INF [19], which provides a distinct scenario from the outdoor large-scale KITTI scenes.

1) Imeplementation Details: For these indoor scenes, we processed the data as follows: First, we stacked multiple LiDAR scans using their provided poses. The combined point cloud was then downsampled with 2cm voxels, and these downsampled points served as the initial positions for Gaussian splats. Each splat was initialized with a radius of 1cm. The first stage, LiDAR-supervised 2DGS geometry construction, required 1000 iterations and took approximately 31±1 seconds. The second stage, the calibration process, needed 3000 iterations and took about 2.8±0.1 minutes. All experiments were conducted using an RTX4090 GPU.

The dataset contains 30 frames of synchronized LiDAR scans and camera images. Since all data were captured with static poses, there are no motion distortion effects to consider. We used the reference LiDAR poses provided by [19], and all camera images were undistorted to follow the perspective camera model. Following the setup in [19], we initialized our algorithm by assuming the LiDAR and camera were at the same pose.

2) Experimental results: The quantitative evaluation of our method is presented in Table IV, which shows the LiDAR-camera extrinsic calibration errors for two indoor scenes. When compared with INF, our method achieves similar levels of accuracy across both translation and rotation parameters. It's important to note that the reference extrinsic parameters themselves have an inherent precision limit of 1-2 cm. Given this measurement uncertainty in the ground truth, the performance of both methods demonstrates satisfactory calibration accuracy for indoor scenarios.

We also provide qualitative results in Fig. 12 to visually demonstrate our calibration performance. In particular, the reprojection results shown in Fig. 12 (c) and (d) compare our calibrated parameters with the reference calibration. The close visual alignment between LiDAR points and image features confirms the accuracy of our calibration results. In Fig. 12 (a) and (b), we present the rendered depth maps and color images from our 2DGS representation. While these renderings show some empty spaces due to the inherent limitations of LiDAR data - specifically, scan sparsity and occlusions - these visualization artifacts do not affect the calibration accuracy.

As our primary goal is precise sensor calibration rather than high-quality rendering, we have not implemented additional techniques to fill these gaps in the visualization.