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

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

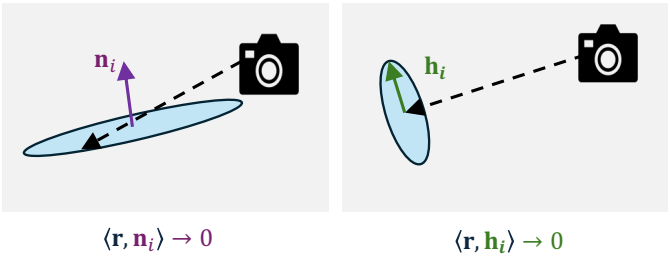


Fig. 7. An illustration of dot product of camera ray with $\mathbf{h}_i$ or $\mathbf{n}_i$.

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

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

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:

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

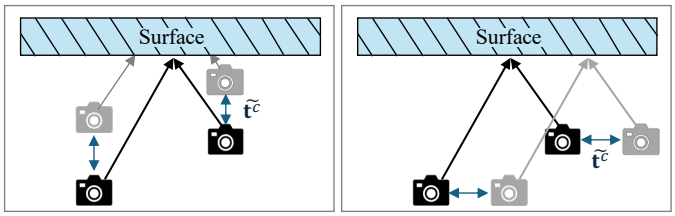
Finally, we can derive Eq. (19):

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

As shown in Fig. 7, Gaussian splats that are perpendicular to the camera ray yield smaller values of $\frac{\langle \mathbf{R}^C \mathbf{r}, \mathbf{h}_i \rangle}{\langle \mathbf{R}^C \mathbf{r}, \mathbf{n}_i \rangle}$. 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:

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

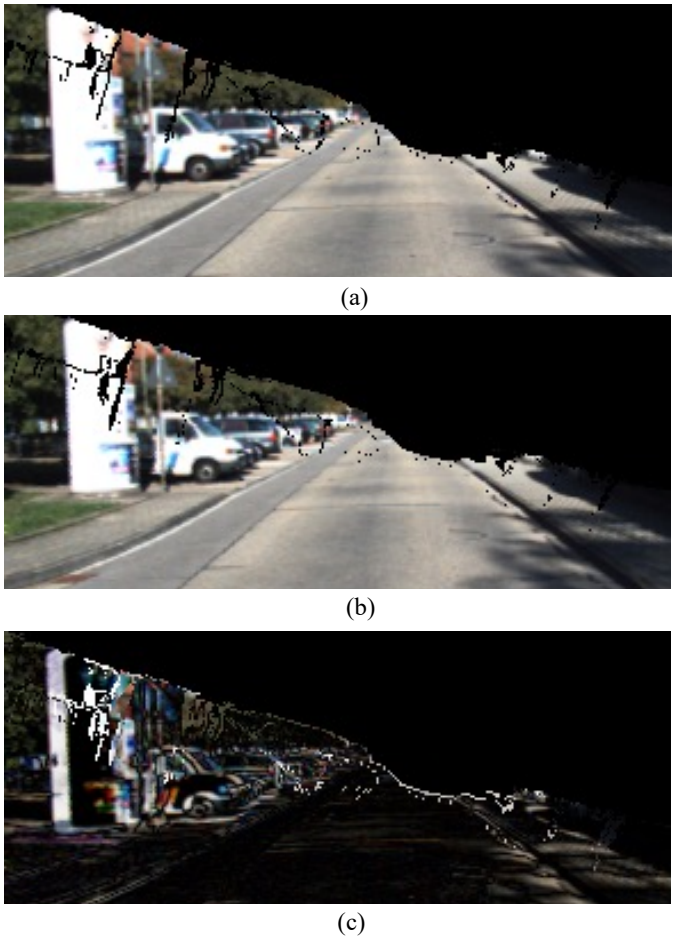
In most cases, these two incorrect rays will not intersect each other on the original Gaussian splat, leading to incorrect color updates for the Gaussian splats. This results in the colors of the Gaussian splats becoming blurred due to the displacement of the camera rays, causing photometric loss. The 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:

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



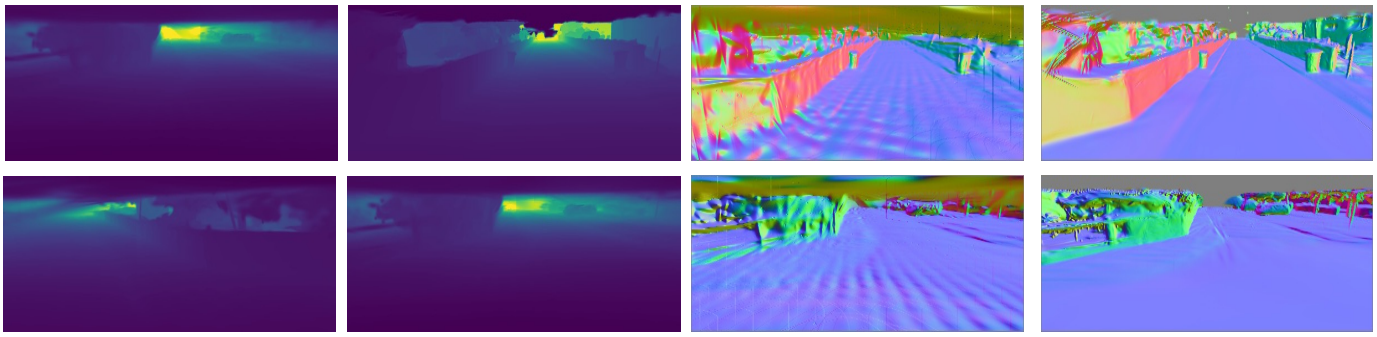
(a) (b) (c)

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 displacements for



(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

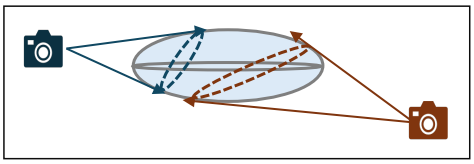


Fig. 11. An illustration of the camera viewed Gaussian splat.

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**

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 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 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

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

(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.

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:

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

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 viewpoints, 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.