# Self-Calibrating Gaussian Splatting for Large Field-of-View Reconstruction

Supplementary Material

## Content



## 1. Supplementary Video

We provide a video, "supp_video.mp4," to better compare our method with baselines. Our video is organized into three parts.

The first part presents a comparison between our method and baselines on the FisheyeNeRF dataset [6] across three scenes. Vanilla 3DGS [7] completely fails to reconstruct the scenes because lens distortion is not accounted for. Fisheye-GS [10] adopts a parametric model, but the peripheral regions produce blurry results, as highlighted by the red box in the corners. In contrast, our method achieves clean and sharp reconstructions.

The second part of the video shows reconstruction results using our method on walk-around captures, including both synthetic and real-world scenes. Our approach achieves sharp and clean renderings upon completing the reconstruction.

The third part of the video compares visualizations of our method with a conventional reconstruction pipeline that either uses narrow-FOV perspective images or undistorted images from COLMAP [14] as input. Our method successfully reconstructs larger regions, particularly for scenes captured with large 180° cameras. Besides, we render videos in fisheye views for each scene.

Additionally, we provide "opt_pose.mp4," which illustrates the camera changes during optimization, and "fisheye-gs_failure.mp4," which demonstrates failure cases of Fisheye-GS [10] in extremely large-FOV settings.

We strongly encourage all reviewers to watch the provided video for a more comprehensive visual understanding of our results.

## 2. Optimization of Camera Parameters

In this section, we first derive the gradients for all camera parameters during training in Sec. 2.1. We then demonstrate the effectiveness of the joint optimization of distortion alongside extrinsic and intrinsic parameters in Sec. 2.2. Finally, we evaluate our camera optimization on synthetic NeRF [12] scenes with significant noise in Sec. 2.3.

### 2.1. Derivation of Gradient

We first derive how to compute the gradient for a pinhole camera and then extend the derivation to account for distortion.

As defined above, the gradient of camera parameters can be written as:

$$\frac{\partial \mathcal{L}}{\partial \Theta} = \sum_{j=1}^{|G|} \frac{\partial \mathcal{L}}{\partial C_j} \frac{\partial C_j}{\partial \Theta} + \frac{\partial \mathcal{L}}{\partial \mu_j^{2D}} \frac{\partial \mu_j^{2D}}{\partial \Theta} + \frac{\partial \mathcal{L}}{\partial \Sigma_j^{2D}} \frac{\partial \Sigma_j^{2D}}{\partial \Theta}. \tag{1}$$

For the color term, we first define the input of the spherical harmonics stored by each Gaussian as the view direction. The 3D location of a Gaussian in world coordinates is $\mathbf{X}_w = [x, y, z]^T$, and the camera center is $\mathbf{C} = [C_x, C_y, C_z]^T$:

$$\omega_j = [x - C_x, y - C_y, z - C_z]^T. \tag{2}$$

Then, the derivative with respect to the camera center is:

$$\sum_{j=1}^{|G|} \frac{\partial \mathcal{L}}{\partial C_j} \frac{\partial C_j}{\partial \Theta} = \sum_{j=1}^{|G|} \frac{\partial \mathcal{L}}{\partial C_j} \frac{\partial C_j}{\partial \omega_j} \frac{\partial \omega_j}{\partial \mathbf{C}} \tag{3}$$

$$= \sum_{j=1}^{|G|} \frac{\partial \mathcal{L}}{\partial C_j} \frac{\partial C_j}{\partial \omega_j} (-\frac{\partial \omega_j}{\partial \mathbf{X}}). \tag{4}$$

For now, we only compute the derivative with respect to the camera center. However, the gradient can be easily propagated back since the camera center corresponds to the translation vector of the inverse of the world-to-camera matrix.

Next, we derive the gradient from the projected 2D Gaussian position $\mu_j^{2D}$. For simplicity, we integrate the intrinsic and extrinsic parameters into a projection matrix $\mathbf{P}$ since we do not yet consider distortion. The projection from a 3D Gaussian in world coordinates is defined by:

$$[\tilde{x}, \tilde{y}, \tilde{z}, \tilde{\omega}]^T = \mathbf{P}[x, y, z, 1]^T. \tag{5}$$

We apply a projective transformation:

$$
\begin{aligned}
f(x,y,z) = x' = \frac{\tilde{x}}{\tilde{\omega}} &= \frac{p_0 x + p_4 y + p_8 z + p_{12}}{p_3 x + p_7 y + p_{11} z + p_{15}}, \\
g(x,y,z) = y' = \frac{\tilde{y}}{\tilde{\omega}} &= \frac{p_1 x + p_5 y + p_9 z + p_{13}}{p_3 x + p_7 y + p_{11} z + p_{15}}, \\
z' = \frac{\tilde{z}}{\tilde{\omega}} &= \frac{p_2 x + p_6 y + p_{10} z + p_{14}}{p_3 x + p_7 y + p_{11} z + p_{15}},
\end{aligned} \tag{6}
$$

where

$$
\mathbf{P} = \begin{bmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{bmatrix}. \tag{7}
$$

Then, we project from NDC to screen space to obtain the 2D location $\mu^{2D} = (u, v)$:

$$
u = \frac{(f(x,y,z)+1)W - 1}{2}, \tag{8}
$$

$$
v = \frac{(g(x,y,z)+1)H - 1}{2}. \tag{9}
$$

The second term can be represented as:

$$
\frac{\partial \mathcal{L}}{\partial \mu_j^{2D}} \frac{\partial \mu_j^{2D}}{\partial \Theta} = \frac{\partial \mathcal{L}}{\partial \mu_j^{2D}} \frac{\partial \mu_j^{2D}}{\partial [x', y']^T} \frac{\partial [x', y']^T}{\partial \mathbf{P}}. \tag{10}
$$

We compute the gradients for each $p_i$ in the projection matrix $\mathbf{P}$:

$$
\begin{aligned}
\frac{\partial L}{\partial p_0} &= \frac{\partial L}{\partial u} \frac{W}{2} \frac{x}{\tilde{\omega}}, \\
\frac{\partial L}{\partial p_4} &= \frac{\partial L}{\partial u} \frac{W}{2} \frac{y}{\tilde{\omega}}, \\
\frac{\partial L}{\partial p_8} &= \frac{\partial L}{\partial u} \frac{W}{2} \frac{z}{\tilde{\omega}}, \\
\frac{\partial L}{\partial p_{12}} &= \frac{\partial L}{\partial u} \frac{W}{2} \frac{1}{\tilde{\omega}}, \\
\frac{\partial L}{\partial p_1} &= \frac{\partial L}{\partial v} \frac{H}{2} \frac{x}{\tilde{\omega}}, \\
\frac{\partial L}{\partial p_5} &= \frac{\partial L}{\partial v} \frac{H}{2} \frac{y}{\tilde{\omega}}, \\
\frac{\partial L}{\partial p_9} &= \frac{\partial L}{\partial v} \frac{H}{2} \frac{z}{\tilde{\omega}}, \\
\frac{\partial L}{\partial p_{13}} &= \frac{\partial L}{\partial v} \frac{H}{2} \frac{1}{\tilde{\omega}},
\end{aligned} \tag{11}
$$

and

$$
\begin{aligned}
\frac{\partial L}{\partial p_3} &= \frac{\partial L}{\partial u}(-x') \left( \frac{W}{2} \frac{x}{\tilde{\omega}} \right) + \frac{\partial L}{\partial v}(-y') \left( \frac{H}{2} \frac{x}{\tilde{\omega}} \right), \\
\frac{\partial L}{\partial p_7} &= \frac{\partial L}{\partial u} \frac{-\tilde{x}}{\tilde{\omega}} \left( \frac{W}{2} \frac{y}{\tilde{\omega}} \right) + \frac{\partial L}{\partial v} \frac{-\tilde{y}}{\tilde{\omega}} \left( \frac{H}{2} \frac{y}{\tilde{\omega}} \right), \\
\frac{\partial L}{\partial p_{11}} &= \frac{\partial L}{\partial u} \frac{-\tilde{x}}{\tilde{\omega}} \left( \frac{W}{2} \frac{z}{\tilde{\omega}} \right) + \frac{\partial L}{\partial v} \frac{-\tilde{y}}{\tilde{\omega}} \left( \frac{H}{2} \frac{z}{\tilde{\omega}} \right), \\
\frac{\partial L}{\partial p_{15}} &= \frac{\partial L}{\partial u} \frac{W}{2} \frac{-\tilde{x}}{\tilde{\omega}^2} + \frac{\partial L}{\partial v} \frac{H}{2} \frac{-\tilde{y}}{\tilde{\omega}^2}.
\end{aligned} \tag{12}
$$

The gradient flow back to intrinsic and extrinsic parameters can be easily computed since $\mathbf{P} = K[R|t]$.

Finally, we compute the last term. The 2D covariance $\Sigma^{2D}$ depends only on the view matrix, so instead of using $\mathbf{P}$, we use only the view matrix (*i.e.*, the world-to-camera matrix) $\mathbf{V}$, and the transformed 3D location in camera coordinates is $\mathbf{X}_c = [x_c, y_c, z_c]^T$:

$$
\begin{aligned}
[x_c, y_c, z_c]^T &= \mathbf{V}[x, y, z, 1]^T \\
&= \begin{bmatrix} v_0 & v_4 & v_8 & v_{12} \\ v_1 & v_5 & v_9 & v_{13} \\ v_2 & v_6 & v_{10} & v_{14} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.
\end{aligned} \tag{13}
$$

We compute the 2D covariance as follows, given known focal lengths $f_x$ and $f_y$:

$$
\begin{aligned}
J &= \begin{bmatrix} J_{00} & J_{01} & J_{02} \\ J_{10} & J_{11} & J_{12} \\ J_{20} & J_{21} & J_{22} \end{bmatrix} = \begin{bmatrix} \frac{f_x}{z_c} & 0 & \frac{-f_x \cdot x_c}{z_c^2} \\ 0 & \frac{f_y}{z_c} & \frac{-f_y \cdot y_c}{z_c^2} \\ 0 & 0 & 0 \end{bmatrix}, \\
\mathbf{W} &= \begin{bmatrix} v_0 & v_4 & v_8 \\ v_1 & v_5 & v_9 \\ v_2 & v_6 & v_{10} \end{bmatrix} = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}, \\
\mathbf{T} &= \mathbf{W} \cdot J = \begin{bmatrix} T_{00} & T_{10} & T_{20} \\ T_{01} & T_{11} & T_{21} \\ T_{02} & T_{12} & T_{22} \end{bmatrix}, \\
\Sigma^{2D} &= \mathbf{T}^T \cdot \Sigma \cdot \mathbf{T} = \mathbf{T}^T \cdot \begin{bmatrix} c_0 & c_1 & c_2 \\ c_1 & c_3 & c_4 \\ c_2 & c_4 & c_5 \end{bmatrix} \cdot \mathbf{T}, \\
a &= \Sigma^{2D}[0][0], \quad b = \Sigma^{2D}[0][1], \quad c = \Sigma^{2D}[1][1], \\
\textbf{Conic} &= \begin{bmatrix} a & b \\ b & c \end{bmatrix}^{-1} = \begin{bmatrix} \frac{c}{ac-b^2} & \frac{-b}{ac-b^2} \\ \frac{-b}{ac-b^2} & \frac{a}{ac-b^2} \end{bmatrix}.
\end{aligned} \tag{14}
$$

Since we only extract the upper three elements of $\Sigma^{2D}$, we do not compute gradients for $T_{20}$, $T_{21}$, and $T_{22}$.

$$
\begin{aligned}
\frac{\partial L}{\partial v_0} =& \frac{\partial L}{\partial T_{00}} \frac{f_x}{z_c}, \quad \frac{\partial L}{\partial v_1} = \frac{\partial L}{\partial T_{01}} \frac{f_x}{z_c}, \\
\frac{\partial L}{\partial v_4} =& \frac{\partial L}{\partial T_{10}} \frac{f_y}{z_c}, \quad \frac{\partial L}{\partial v_5} = \frac{\partial L}{\partial T_{11}} \frac{f_y}{z_c}, \\
\frac{\partial L}{\partial v_2} =& \frac{-(v_0 \frac{\partial L}{\partial T_{00}} + v_1 \frac{\partial L}{\partial T_{01}}) \cdot x \cdot f_x}{z_c^2} \\
&+ \frac{-(v_4 \frac{\partial L}{\partial T_{10}} + v_5 \frac{\partial L}{\partial T_{11}} + v_6 \frac{\partial L}{\partial T_{12}}) \cdot x \cdot f_y}{z_c^2} \\
&+ \frac{(z_c - v_2 x) f_x \frac{\partial L}{\partial T_{02}}}{z_c^2}, \\
\frac{\partial L}{\partial v_6} =& \frac{-(v_0 \frac{\partial L}{\partial T_{00}} + v_1 \frac{\partial L}{\partial T_{01}} + v_2 \frac{\partial L}{\partial T_{02}}) \cdot y \cdot f_x}{z_c^2} \\
&+ \frac{-(v_4 \frac{\partial L}{\partial T_{10}} + v_5 \frac{\partial L}{\partial T_{11}}) \cdot y \cdot f_y}{z_c^2} \\
&+ \frac{(z_c - v_6 y) f_y \frac{\partial L}{\partial T_{12}}}{z_c^2}, \\
\frac{\partial L}{\partial v_{10}} =& \frac{-((v_0 \frac{\partial L}{\partial T_{00}} + v_1 \frac{\partial L}{\partial T_{01}} + v_2 \frac{\partial L}{\partial T_{02}}) f_x}{z_c^2} \\
&+ \frac{(v_4 \frac{\partial L}{\partial T_{10}} + v_5 \frac{\partial L}{\partial T_{11}} + v_6 \frac{\partial L}{\partial T_{12}}) f_y) z}{z_c^2}, \\
\frac{\partial L}{\partial v_{14}} =& \frac{-((v_0 \frac{\partial L}{\partial T_{00}} + v_1 \frac{\partial L}{\partial T_{01}} + v_2 \frac{\partial L}{\partial T_{02}}) f_x}{z_c^2} \\
&+ \frac{(v_4 \frac{\partial L}{\partial T_{10}} + v_5 \frac{\partial L}{\partial T_{11}} + v_6 \frac{\partial L}{\partial T_{12}}) f_y}{z_c^2}.
\end{aligned} \tag{15}
$$

To account for distortion, we decompose the projection matrix into two separate operations. Following the definitions above, we apply an invertible ResNet in between:

$$
[\tilde{x}, \tilde{y}, \tilde{z}, \tilde{\omega}]^T = K \, \text{homo}(z_c \cdot \text{homo}(\mathcal{D}_\theta \, (\text{proj}(x_c, y_c, z_c)))). \tag{16}
$$

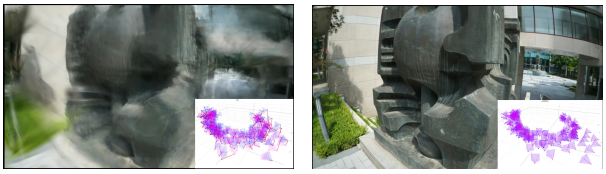
For the projection of the Gaussian covariance $\Sigma$, we compute a new Jacobian matrix $J$ using the distorted $\mathbf{X}_c$:

$$
[x_c, y_c, z_c]^T = z_c \cdot \text{homo}(\mathcal{D}_\theta \, (\text{proj}(x_c, y_c, z_c))). \tag{17}
$$

Both processes are differentiable, allowing us to compute the intermediate Jacobian for $\mathcal{D}_\theta$ to update the invertible ResNet.

### 2.2. Joint Distortion and Pose Optimization

Our approach supports efficient optimization of camera parameters, either independently (with perspective images) or in combination, as shown in Fig. 1, with distortion modeling. This ensures that our pipeline remains robust even when both distortion and camera parameters are inaccurate.



(a) w/o Cameras Optimization (b) w/ Cameras Optimization

Figure 1. **Camera Parameters Optimization**. We initialize noisy cameras from COLMAP [14]. (a) Modeling fisheye distortion without optimizing camera parameters, while (b) jointly optimizing both in a self-calibration manner. Our full model can recover accurate lens distortion and camera parameters simultaneously.

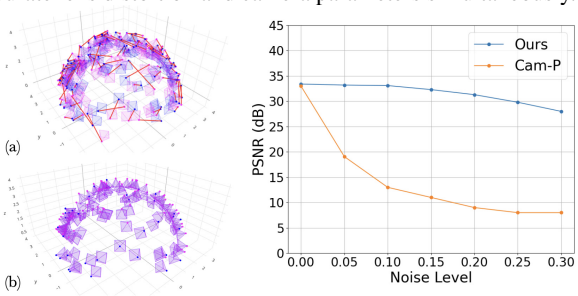
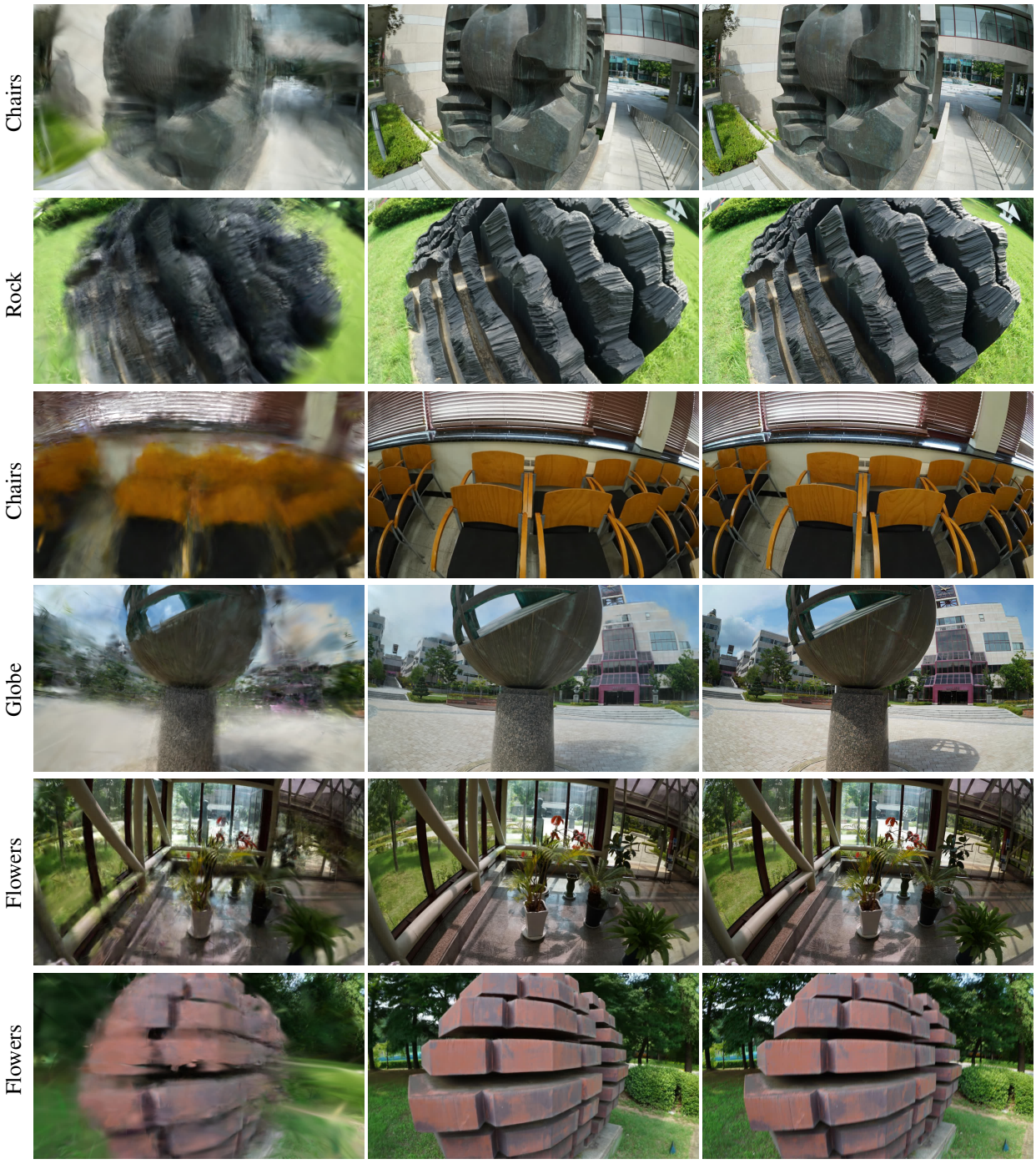



Figure 2. Visual comparison of (a) the initial perturbed ($s = 0.15$) and GT poses and (b) optimized camera poses in the Lego scene. The chart demonstrates the different level of perturbations and the effectiveness of our optimization. Our method successfully recovers accurate camera frames.

To evaluate our model's ability to jointly optimize lens distortion and other camera parameters in real-world scenes, we introduce a setting where both camera extrinsics and intrinsics are perturbed in the FisheyeNeRF dataset. Specifically, we add Gaussian noise with a standard deviation of $s = 0.15$ to each image's camera extrinsic and intrinsic parameters. Despite this additional noise, our method successfully recovers the lens distortions while generating high-quality novel-view synthesis renderings (Fig. 3).

One key observation is that COLMAP [14] can robustly estimate camera extrinsics but struggles with intrinsic parameters and distortion. When we enable extrinsic optimization during reconstruction, the camera poses are refined only slightly, indicating that the initial poses are already quite reliable. Regarding lens distortion, as demonstrated in Fig. 5 and discussed in **Hybrid Field** of Sec.4.4 andSec. 3, COLMAP's distortion estimation lacks accuracy, highlighting the necessity of our hybrid distortion field for improved expressiveness and precision.

We hypothesize that this limitation stems from the structure-from-motion (SfM) pipeline [14] in COLMAP. COLMAP primarily utilizes the central region of raw images (corresponding to a small FOV), where conventional distortion models perform well. Consequently, lens distortion has minimal influence on extrinsic estimation, as COLMAP can still rely on the image center to solve linear




(a) w/o Optimize Cameras (b) Optimize Cameras (c) GT

Figure 3. **Qualitative Comparison on Perturbed FisheyeNeRF dataset [6]**. We show the novel view rendering with perturbed camera poses. We disable and enable camera optimization to illustrate the capability of our pipeline on recovering inaccurate poses along with distortion modeling.

equations. However, when attempting to leverage the full FOV of raw images for reconstruction, the limitations of a fixed distortion model and a single-plane projection become pronounced.

To validate the self-calibration capability of our pipeline, we manually introduce noise into the extrinsics produced

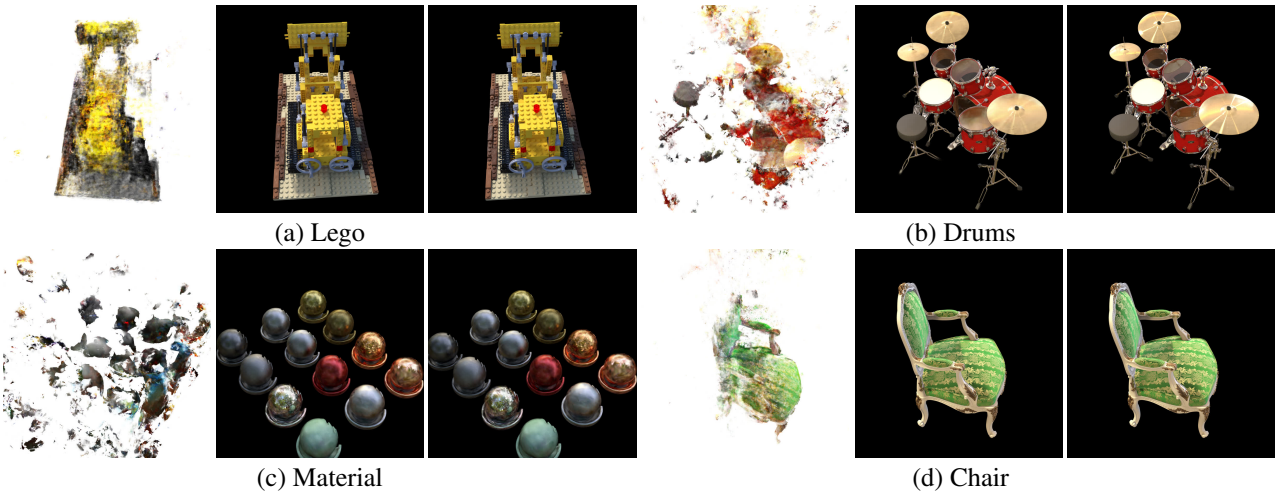



Figure 4. We carry qualitative comparison with CamP at noise level 0.15. Each scene show CamP, our method, and the ground truth, from left to right. Our method is able to produce sharp renderings at this noise level, where CamP fails.

| Methods | Image Metrics | | | Camera Metrics | |
|---|---|---|---|---|---|
| | PSNR | SSIM | LPIPS | Position | Orientation |
| 3DGS | 16.54 | 0.733 | 0.273 | 0.2911 | 5.015 |
| CamP | 19.07 | 0.840 | 0.289 | 0.1879 | 5.619 |
| Ours | **32.84** | **0.964** | **0.034** | **0.0082** | **0.919** |

Table 1. Comparison with CamP [13] and 3DGS [7] in the NeRF-Synthetic dataset. We report average camera orientation errors in degrees, and position error in world units.

by COLMAP and jointly optimize extrinsics, intrinsics, and distortion. As shown in Fig. 3, our approach effectively refines all camera parameters and distortion simultaneously. Without extrinsic and intrinsic optimization, the hybrid field can only predict coarse distortion, while misaligned poses contribute to the blurry reconstruction seen in Fig. 3 (a). Even with significant perturbations in poses and intrinsics, our method robustly recovers accurate camera parameters and the distortion field after training, as illustrated in Fig. 3 (b).

### 2.3. Pure Camera Optimization

We use the NeRF-Synthetic dataset [12], which includes known ground truth camera poses. The dataset contains 100 viewpoints for training and 200 additional viewpoints for computing test error metrics. We first add noise to perturb the rotation angles of camera extrinsics, the positions of the camera centers, and the focal lengths. These noisy cameras are used to train both the baselines and our method. We compare our method with CamP [13], implemented on Zip-NeRF [3], a state-of-the-art method for joint optimization of 3D scenes and camera extrinsics and intrinsics. In addition to CamP, we also report the performance of vanilla Gaussian Splatting without pose optimization.

| **Scenes** | **Metric** | **0** | **0.1** | **0.15** | **0.2** | **0.25** |
|---|---|---|---|---|---|---|
| Chair | SSIM | 0.987 | 0.988 | 0.988 | 0.987 | 0.980 |
| | PSNR | 35.81 | 36.10 | 35.99 | 35.83 | 34.35 |
| | LPIPS | 0.012 | 0.012 | 0.012 | 0.012 | 0.019 |
| Lego | SSIM | 0.983 | 0.974 | 0.972 | 0.969 | 0.965 |
| | PSNR | 35.77 | 34.78 | 34.29 | 33.70 | 33.15 |
| | LPIPS | 0.015 | 0.021 | 0.023 | 0.025 | 0.029 |
| Drums | SSIM | 0.955 | 0.954 | 0.953 | 0.953 | 0.946 |
| | PSNR | 26.17 | 26.15 | 26.04 | 26.01 | 25.30 |
| | LPIPS | 0.037 | 0.038 | 0.039 | 0.040 | 0.045 |
| Materials | SSIM | 0.960 | 0.960 | 0.951 | 0.942 | 0.843 |
| | PSNR | 29.99 | 29.93 | 28.91 | 27.95 | 15.17 |
| | LPIPS | 0.034 | 0.036 | 0.044 | 0.052 | 0.158 |
| Mic | SSIM | 0.991 | 0.989 | 0.987 | 0.974 | 0.923 |
| | PSNR | 35.34 | 34.58 | 33.65 | 30.07 | 18.78 |
| | LPIPS | 0.006 | 0.008 | 0.010 | 0.019 | 0.087 |
| Ship | SSIM | 0.907 | 0.873 | 0.776 | 0.719 | 0.700 |
| | PSNR | 30.91 | 28.66 | 20.96 | 16.80 | 15.10 |
| | LPIPS | 0.106 | 0.126 | 0.203 | 0.262 | 0.294 |
| Ficus | SSIM | 0.987 | 0.987 | 0.984 | 0.955 | 0.859 |
| | PSNR | 34.85 | 34.84 | 33.99 | 28.08 | 18.54 |
| | LPIPS | 0.012 | 0.012 | 0.014 | 0.039 | 0.125 |
| Hotdog | SSIM | 0.985 | 0.985 | 0.985 | 0.983 | 0.982 |
| | PSNR | 37.67 | 37.65 | 37.63 | 37.05 | 36.53 |
| | LPIPS | 0.020 | 0.020 | 0.020 | 0.022 | 0.025 |

Table 2. Quantitative Comparison on the Perturbed Synthetic Dataset

The models are evaluated on two criteria, following the protocol in CamP [13]. First, we measure the accuracy of

| Method | Chairs | | | Cube | | | Flowers | | | Globe | | | Heart | | | Rock | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS |
| 3DGS [7] | 0.431 | 14.06 | 0.547 | 0.507 | 15.21 | 0.533 | 0.281 | 12.91 | 0.609 | 0.502 | 15.09 | 0.530 | 0.505 | 15.19 | 0.549 | 0.297 | 12.70 | 0.595 |
| Ours (Freeze Init) | 0.583 | 18.28 | 0.290 | 0.637 | 21.64 | 0.296 | 0.443 | 18.09 | 0.379 | 0.580 | 19.63 | 0.327 | 0.660 | 20.87 | 0.282 | 0.511 | 20.24 | 0.280 |
| Ours | **0.832** | **23.45** | **0.106** | **0.786** | **24.63** | **0.162** | **0.693** | **22.01** | **0.172** | **0.790** | **23.63** | **0.126** | **0.775** | **23.42** | **0.195** | **0.787** | **24.88** | **0.145** |

Table 3. **Ablation on Invertible ResNet Optimization**. We compare our final optimized hybrid field with a fixed hybrid field reconstruction on FisheyeNeRF [6]. While the distortion estimated from COLMAP [14] significantly improves quality compared to vanilla 3DGS [7], optimizing the invertible ResNet further enhances performance.


(a) Freeze Intialization (b) Ours (c) GT

Figure 5. **Comparison on Invertible ResNet Optimization**. This version shows the comparison transposed, grouping by scenes instead of optimization stages.

the estimated camera poses in the training views after accounting for a global rigid transformation. Second, we measure the quality of rendered images at the held-out camera poses after a test-time optimization to adjust for a potential global offset. Tab. 1 shows that our method outperforms both vanilla 3DGS and CamP in both image and camera metrics by a large margin.

We further evaluate our model by perturbing the camera poses with varying levels of noise. Specifically, we add Gaussian noise with a standard deviation ranging from 0 to 0.3 to the camera poses. For reference, we report the increased noise levels for each scene in Tab. 2. Handling larger noise in camera poses presents a significant challenge. As shown in Fig. 2 and Fig. 4, CamP's performance drops significantly due to its tendency to fall into local minima when the initial camera poses contain substantial errors. In contrast, our method exhibits much slower degradation in novel-view synthesis performance.

We also include the video "pose_opt.mp4" to visualize the optimization process.

## 3. Distortion Estimation from COLMAP

In practice, the distortion estimated from the SfM [14] pipeline can be used as an initialization for our hybrid field, stabilizing training and accelerating convergence. However, these parameters are inaccurate when derived from highly distorted images. We verify the necessity of optimizing our hybrid field during reconstruction both quantitatively and qualitatively.

To assess this, we fit the COLMAP [14] distortion parameters within our hybrid field and freeze the network. As shown in Tab. 3, the distortion initialization from COLMAP improves upon Vanilla 3DGS [7] but remains far from accurate in modeling distortion compared to our approach.

(a) Low Resolution (b) High Resolution

Figure 6. **Resolution of Control Grid**. When the resolution of the control grid is decreased, the central region retains decent quality due to minimal distortion. However, as highlighted by the red and blue boxes in the corners of the image, a sparse control grid for the hybrid field results in noticeably distorted renderings.

While the hybrid field produces a roughly correct distortion field, it results in blurry reconstructions, as shown in Fig. 5. This degradation is particularly noticeable in scenes with many straight lines, such as the jalousie windows in the chairs scene. The photometric evaluation in Tab. 3 demonstrates that our fully optimized hybrid field significantly outperforms the estimated distortion parameters. These results highlight the importance of optimizing our hybrid field for achieving more accurate reconstruction and distortion modeling.

## 4. Computational Efficiency

**Training Time.** To verify the hypothesis that our hybrid method achieves a better balance between expressiveness and efficiency, we perform an ablation study on the FisheyeNeRF dataset [6]. Specifically, we analyze the grid resolution of $\mathbf{P}_c$, which determines the number of evaluations required for the most computationally expensive part of the pipeline—the invertible ResNet. Tab. 4 reports the PSNR and training time for three different grid resolutions, as well as for Fisheye-GS [10] and vanilla 3DGS [7]. We observe that increasing the resolution of $\mathbf{P}_c$ leads to better performance but longer training time. Further reducing the grid resolution does not significantly shorten computation time, as other operations, such as gradient computation for camera parameters, begin to dominate. Fortunately, all hybrid solutions significantly outperform vanilla 3DGS and Fisheye-GS. The lowest-resolution setting we tested introduces only a 7-minute training time overhead compared to 3DGS, in exchange for a $> 8$ dB boost in PSNR.

**Control Grid Resolution.** A higher-resolution control grid results in a smoother distortion field representation but slower training. To better illustrate the effect of control grid resolution, we visualize two types of resolutions (*i.e.,* $265 \times 149$ and $66 \times 37$). While there are no significant differences in the central region, the distortion at the edges is better recovered with a higher-resolution grid. Since the distortion field becomes smoother, a high-resolution control grid produces more accurate distorted lines, as shown in the red and blue boxes of Fig. 6.

| $\mathbf{P}_c$ Resolution | PSNR (↑) | Time (mins) |
|---|---|---|
| $265 \times 149$ | 23.67 | 55 |
| $132 \times 74$ | 22.99 | 36 |
| $66 \times 37$ | 22.44 | 25 |
| Explicit Grid | 14.93 | 22 |
| Fisheye-GS [10] | 21.84 | 46 |
| 3DGS [7] | 14.19 | 18 |

Table 4. **Ablation Study on Control Grid Resolution**. Parameter study on different control point resolutions, showing that our method has favorable cost/performance trade-off.

| Method | Garden | | | Studio | | |
|---|---|---|---|---|---|---|
| | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS |
| Fisheye-GS [10] | 0.530 | 14.94 | 0.542 | 0.536 | 12.24 | 0.549 |
| Ours | **0.882** | **27.85** | **0.144** | **0.965** | **33.86** | **0.044** |

Table 5. **Evaluation on Real-World Wide-Angle Captures**. We evaluate Fisheye-GS [10] and our method on two real-world scenes captured with large FOV fisheye cameras. Our method outperforms the baseline by a significant margin.

| Method | Test View | Num | SSIM | PSNR | LPIPS |
|---|---|---|---|---|---|
| Fisheye-GS [10] | Fisheye | 100 | 0.630 | 15.94 | 0.463 |
| Ours | | 100 | **0.886** | **30.72** | **0.146** |

Table 6. **Evaluation on Mitsuba Scenes**. The comparison between our method and Fisheye-GS [10] illustrates the expressiveness of our hybrid field.

## 5. Extra Experiments

### 5.1. Quantitative Comparisons with Fisheye-GS

In addition to Fig. 5 in the main paper, we also provide a quantitative evaluation of our method compared with the baseline Fisheye-GS [10] in Tab. 5 and Tab. 6. The performance degradation observed in the baseline method is primarily due to the limitations of the conventional camera distortion model used during reconstruction, which struggles at the edges of large FOVs. As a result, there is no geometric consistency in the peripheral regions to produce uniform gradients for optimizing the Gaussians. When rendering novel views, these Gaussians appear as large floaters that occlude the camera view. We further visualize this phenomenon in the failure case video reconstructed by Fisheye-GS [10]. The center area is revealed as we gradually decrease the scale of the Gaussians. Please refer to our sup-

(a) Living (b) Kitchen (c) Room

Figure 7. **Single Planar Projection with Hybrid Field**. Our hybrid field can be directly applied to a single plane during rasterization. However, the limitation of single planar projection is that it cannot cover the full FOV of the raw images, leading to partial loss of information in the peripheral regions.

plementary video "fisheye-gs_failure.mp4" for a comparison with the parametric distortion model.

### 5.2. Qualitative Results of Cubemap Ablation

As illustrated in Fig. 2 of the main paper, we apply a cubemap to overcome the limitations of perspective projection. Even with our hybrid distortion field, we can only utilize the central region of raw fisheye images, as shown in the bottom row of Fig. 7. In contrast, rendering a cubemap enables us to achieve a larger FOV, allowing us to compute the photometric loss with reference raw images for optimization, as demonstrated in Fig. 7 of the main paper. We also provide a quantitative evaluation in Tab. 4 of the main paper, showing that the final reconstruction quality is compromised when using a single planar projection.

Additionally, the boundary of the final distorted rendering is irregular, primarily because the distortion information at the boundary heavily relies on extra regions that are not covered by single planar projections.

### 5.3. Qualitative Comparisons with Different Numbers of Input Views

We report quantitative results in Tab. 2 of the main paper, where our method outperforms the baseline even with significantly fewer input views, down to 10-15%.

To illustrate the impact of varying input view counts, we visualize the rendering quality as the number of input views decreases. Even with as few as 25 input views, our method still achieves reasonable performance, as shown in Fig. 8. This demonstrates our method's ability to effectively utilize large FOV cameras and achieve comprehensive scene coverage during reconstruction.

| Method | Num | Mitsuba Kitchen | | | Mitsuba Room1 | | | Mitsuba Room2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS |
| 3DGS [7] | 200 | 0.470 | 11.09 | 0.435 | 0.595 | 15.27 | 0.406 | 0.897 | 30.88 | 0.155 |
| Ours | 10 | 0.644 | 23.03 | 0.346 | 0.556 | 22.26 | 0.377 | 0.644 | 23.03 | 0.346 |
| | 25 | 0.689 | 24.75 | 0.303 | 0.613 | 24.21 | 0.318 | 0.825 | 24.81 | 0.254 |
| | 50 | 0.708 | 25.44 | 0.285 | 0.640 | 25.07 | 0.301 | 0.856 | 28.26 | 0.214 |
| | 100 | **0.794** | **27.56** | **0.272** | **0.686** | **26.27** | **0.314** | **0.920** | **33.21** | **0.107** |
| Fisheye-GS [10] | 100 | 0.601 | 14.30 | 0.485 | 0.549 | 15.54 | 0.548 | 0.739 | 17.97 | 0.355 |
| Ours | 10 | 0.801 | 26.04 | 0.202 | 0.703 | 23.69 | 0.232 | 0.806 | 19.06 | 0.256 |
| | 25 | 0.855 | 28.62 | 0.155 | 0.783 | 26.71 | 0.180 | 0.853 | 24.28 | 0.167 |
| | 50 | 0.870 | 29.55 | 0.151 | 0.801 | 27.57 | 0.172 | 0.883 | 27.29 | 0.133 |
| | 100 | **0.886** | **30.72** | **0.146** | **0.842** | **28.46** | **0.180** | **0.929** | **31.16** | **0.095** |

Table 7. **Evaluation on Mitsuba Synthetic Scenes**. We compare our method with vanilla 3DGS [7] and Fisheye-GS [10] on a set of held-out captures. Since vanilla 3DGS does not support fisheye rendering, we render several perspective images at the same locations and look-at directions for comparison. We directly compare the fisheye rendering results with both Fisheye-GS and our method.

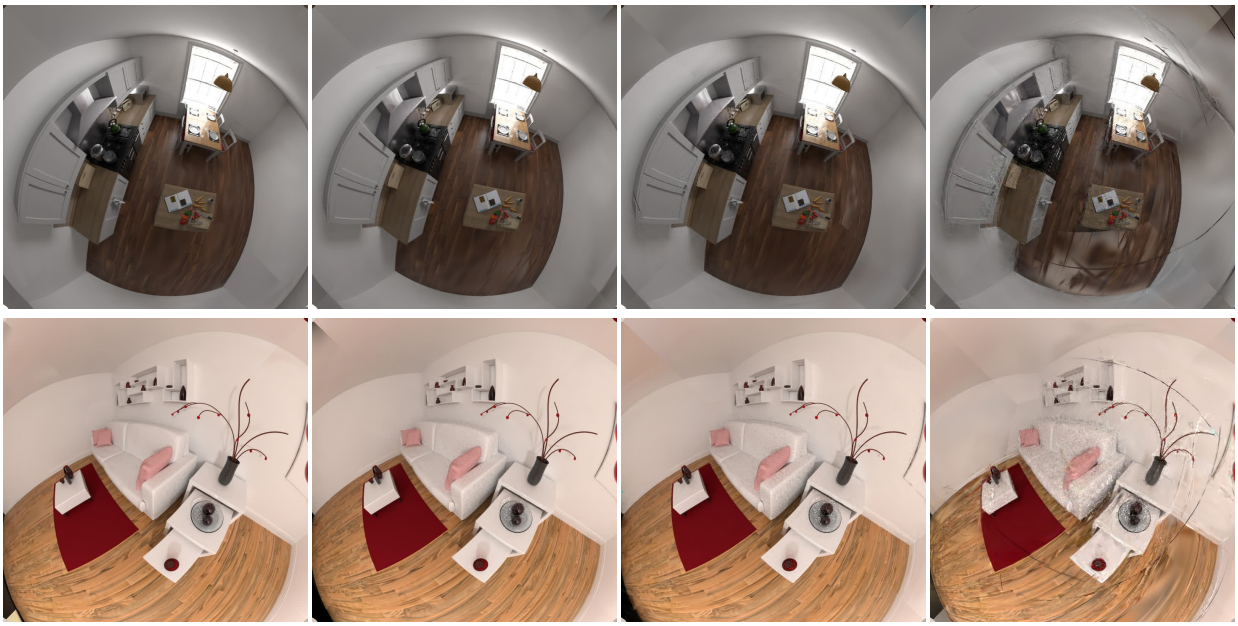
### 5.4. Comparisons with Regular FOV Cameras

**Synthetic Mitsuba Scene Captures.** To carefully control the experimental settings, we customized a camera module in the Mitsuba ray tracer [5] using camera parameters derived from DSLR lenses, as profiled in the open-source Lensfun [1]. We also utilize 3D assets, including geometry, materials, and lighting, from [4] to produce renderings. Our synthetic dataset contains three large indoor scenes and four object-centric scenes. All indoor scenes follow a Sigma 180° circular fisheye camera. Two of the object scenes are rendered with a 120° fisheye lens, while the others are rendered with classic radial distortion.

To capture detailed perspectives of the scene, cameras are placed close to the objects at varying distances. For Room2, since objects are uniformly distributed in the space, we placed a set of camera centers along a Hilbert curve, then oriented each fixed camera center to cover the surroundings. The number of images captured at each point is reduced for 180° images compared to those with a 90° FOV, as shown in the number of captures in Tab. 7.

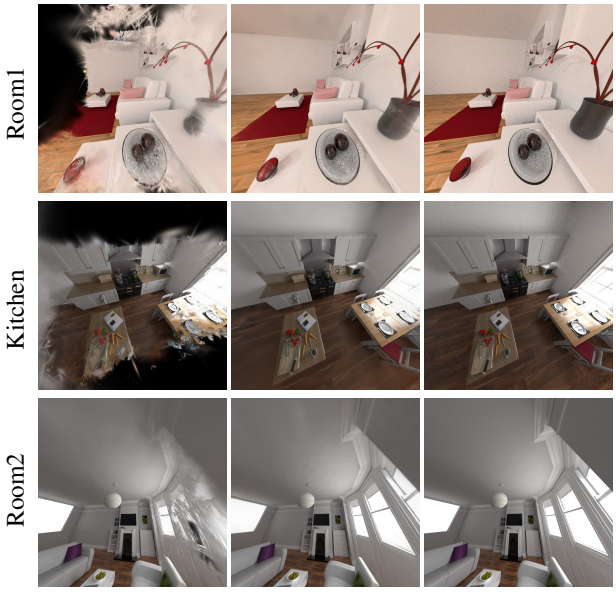
**Qualitative Results.** To verify the accuracy of self-calibration, we generate a set of hold-out cameras that share the same distribution as the training set. For each validation camera, we render paired perspective and 180° fisheye images. Unlike real-world datasets such as Garden and Studio, our synthetic dataset allows direct comparison with ground-truth perspective views. As shown in Fig. 9, we render an additional 20° for the hold-out cameras to highlight the difference in coverage between our method and conventional capture approaches.

It is worth noting that in the comparison between our method and 3DGS [7], we evaluate in perspective views since 3DGS [7] does not support fisheye rendering, whereas our method can generate perspective views after training. In contrast, we directly compare our method with Fisheye-



(a) 100 Views (b) 50 Views (c) 25 Views (d) 10 Views

Figure 8. **Qualitative Evaluation of Reconstruction with Varying Numbers of Large FOV Inputs**. Our method achieves high-quality reconstruction even with a relatively small number of input images, thanks to our hybrid distortion representation and cubemap resampling.




(a) Small FOV (b) Ours (b) GT

Figure 9. **Evaluation of Perspective Rendering**. After reconstruction, our method can render perspective views with arbitrary FOV. We compare the perspective renderings produced by our method with those rendered from small-FOV reconstructions using 3DGS [7].

GS [10], as both methods natively support fisheye rendering.

### 5.5. More Real-world Scenes

**Scene Captures.** We use a Canon 5D Mk III DSLR camera with a Canon 8mm-15mm fisheye lens, zoomed out to 8mm with a 180° FOV, to capture a complex indoor office scene, where we place models and spheres on a table. Images are taken close to the table to capture the details of the various models and spheres. We also use a Meike 3.5mm f/2.8 ultra-wide-angle circular fisheye lens to capture the same office.

Additionally, we mount two fisheye cameras on a rig configured such that the cameras are perpendicular to each other. This camera rig is used to capture a backyard scene by walking clockwise and counterclockwise twice to record videos. The benefit of using this rig is that the relative pose between the two fisheye cameras is fixed, simplifying the SfM [14] pipeline for estimating accurate poses.

**Qualitative Results.** Qualitative results are shown in Fig. 10. Our method effectively recovers details in the central region while accurately modeling lens distortion for background elements, such as painting frames and lines on the white wall.

As shown in Fig. 11, our method converges to an accurate calibration, ensuring that lines on the house's surface and the ladder leaning against the tree remain straight when rendered in perspective views.

(a) Ours (Perspective) (b) Ours (Fisheye) (c) GT

Figure 10. **Reconstruction from 180° FOV Fisheye Captures.** Using a Canon fisheye camera, we capture the scene and reconstruct the office with our method. Both perspective and fisheye views are rendered to demonstrate the quality of our reconstruction.

### 5.6. Adaptability to Different Lens Distortions

As mentioned in the last part of Sec. 4.2 of the main paper and also shown in Fig. 6, we introduced synthetic distortions, including both radial and tangential components, to images from the LLFF dataset [11].

We also apply moderate radial distortion to our synthetic dataset and reconstruct several object scenes. After training, we can render undistorted images. The perspective rendering increases the FOV while maintaining the same camera extrinsics. As shown in Fig. 12, distorted edge lines, such as those on the Lego and Car objects, are correctly recovered into straight lines, demonstrating the capability of our hybrid field to model radial distortion effectively. We also report quantitative evaluations in Tab. 8. Note that the radial distortion is relatively subtle, so the improvement is not as pronounced compared to other types of lenses.

(a) Ours (Perspective) (b) Ours (Fisheye) (c) GT

Figure 11. **Large FOV Reconstruction from a Customized Fisheye Rig.** We reconstruct a backyard from images captured using a fisheye rig. Our method achieves accurate geometric corrections, such as straightening the lines on the wall and the edges of the house.

| Method | PSNR | SSIM | LPIPS |
|---|---|---|---|
| Vanilla-GS [7] | 25.37 | 0.923 | 0.121 |
| Ours | **28.00** | **0.932** | **0.093** |

Table 8. **Evaluation on Radial Distortion of Mitsuba Synthetic Scenes**. We compare our method with 3DGS [7] in object-centric scenes with slight radial distortion, where our method still produces better reconstructions.

### 5.7. Comparison of Regular and Invertible ResNet

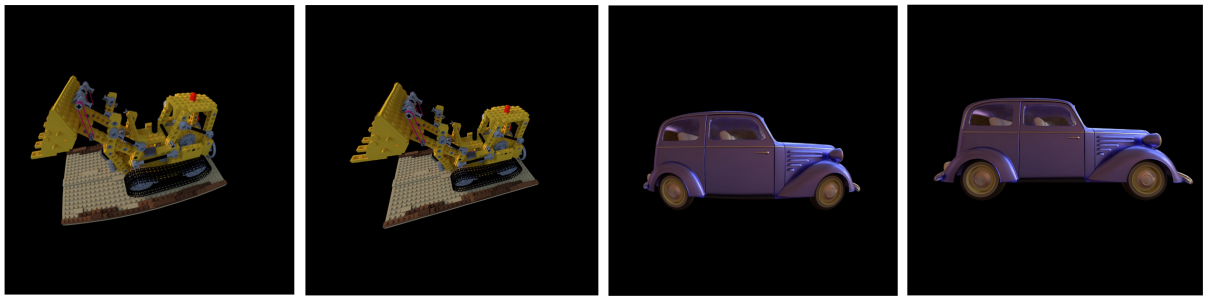
Neural networks can model complex non-linear fields, but the key advantage of iResNet is its effective regularization. Light rays passing through the lens are strictly bijective and invertible. iResNet, using fixed-point iteration, enforces this property at minimal cost. We visualize the error map compared to the distortion of the GT lens in synthetic scenes in Fig. 13. We show that iResNet predicts smooth distortion with low error, whereas ResNet produces a highly asymmetric field with large errors. The large error produced by ResNet is largely due to the lack of regularization. The displacement predicted by ResNet can be arbitrary and does not follow the two properties that real-world light rays hold.

### 5.8. Qualitative Results in Undistorted Rendering

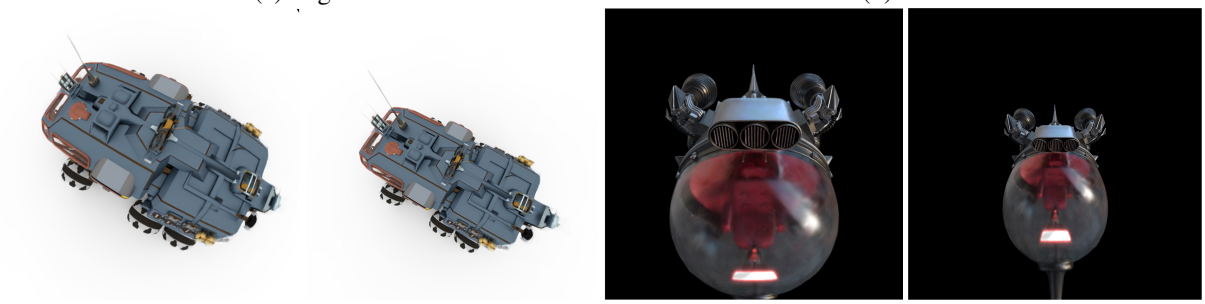
We provide additional rendering results in this section. We render both fisheye and perspective views on the FisheyeNeRF dataset [6]. In Fig. 14, we fix the view direction and camera location for fisheye rendering and extend the FOV for perspective rendering. In scenes such as Cube, Chairs, and Flowers, we observe that straight lines are accurately recovered during reconstruction. The lines on the wall behind the Cube and the window frames serve as strong evidence that our self-calibration system precisely models lens distortion.

## 6. Implementation Details

Our implementation is based on the codebase from Gaussian Splatting [7] and gsplat [15]. We use the same loss function as 3DGS for training [7]. The invertible ResNet is constructed using FrEIA [2]. We follow Kerbl et al. [7] to select hyperparameters for optimizing 3D Gaussians. We also adopt the implementation of MCMC densification [8]. Compared with vanilla densification, MCMC helps remove floaters by using opacity thresholding to relocate dead Gaussians. While the final quantitative results on the



(a) Lego (b) Car



(c) Rover (d) Spaceship

Figure 12. **Radial and Perspective Rendering**. We evaluate our method on a synthetic radial distortion dataset. Our approach successfully recovers slight radial distortion during reconstruction and enables perspective rendering upon completion of training.

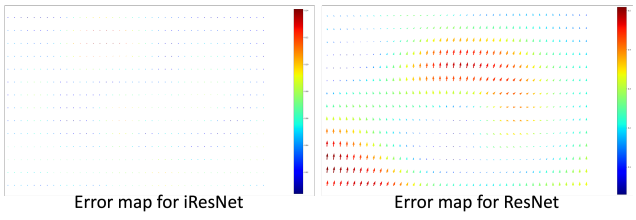



Figure 13. **Distortion Error Map**. We visualize the error map between the predicted distortion and the ground truth distortion from Mitsuba synthetic scenes.

test set remain largely unchanged, applying the MCMC technique reduces visual floaters in novel viewpoints. For high-resolution scene captures such as Backyard and Office, we also use bilateral grids and anti-aliasing [16] for improved quality.

As explained in Fig. 5, optimizing our hybrid field is essential for successful self-calibration. We use Adam [9] to train the invertible ResNet. The initial learning rate for the invertible ResNet is set to 1e-5 and gradually decreases to 1e-7 for FisheyeNeRF [6]. The final learning rate for real-world captures is 1e-8, including Studio, Garden, and Backyard in Figure 11, as well as more complex real-world captures such as Office Fig. 10. The learning rate for Mitsuba indoor synthetic scenes is set to 1e-8, while for object-centric scenes, it is 1e-7. All experiments are conducted on a single NVIDIA GeForce RTX 3090.

## 7. Failure Cases and Limitations

Real-world outdoor captures often include the sky. Reconstructing the sky poses challenges due to moving clouds and the large uniform regions of blue and white without textures. The 3DGS [7] method tends to assign large Gaussians to the sky, resulting in artifacts when rendering novel views. Occasionally, some large Gaussians leak into the scene's center, appearing as a thin film in front of the camera. Similarly, for indoor scenes, regions with uniform textures, such as colored walls, present challenges. These textureless walls are often represented by Gaussians with large covariance matrices, causing similar rendering artifacts as observed with the sky.

The Gaussian sorting we propose alleviates the intensity discontinuities at the boundaries of cubemap faces caused by the multiple projections of a single Gaussian. However, since the projection of 3D covariance follows the equation in Sec. 3.1 of the main paper, identical 3D Gaussians can still result in different 2D covariances on different faces. This issue can be addressed by implementing smoother transitions between projection faces, such as spherical projection.

Finally, we do not account for the entrance pupil shift phenomenon commonly observed in fisheye lenses. This effect is distinct from the lens distortion we are currently modeling. As a result, our method still struggles with such cameras, as shown in Figure 10. While entrance pupil shift

(a) Heart (b) Cube

(c) Flowers (d) Chairs

Figure 14. **Fisheye and Perspective Rendering**. After optimization, our method allows rendering in either fisheye or perspective views. Perspective rendering can be achieved by simply removing the hybrid field.

is negligible for distant scenes, it can cause splat misalignments in near-field scenes (*e.g.,* the blurry sphere surface in the Office scene shown in the video), as the shift can reach up to half a centimeter for full-frame lenses. It remains an exciting direction to study how to model such lens effects to further improve reconstruction quality.

## References


[1] Lensfun. https://lensfun.github.io/. 8

[2] Lynton Ardizzone, Till Bungert, Felix Draxler, Ullrich Köthe, Jakob Kruse, Robert Schmier, and Peter Sorrenson. Framework for Easily Invertible Architectures (FrEIA), 2018-2022. 11

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *ICCV*, 2023. 5

[4] Benedikt Bitterli. Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 8

[5] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba renderer, 2010. 8

[6] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *ICCV*, 2021. 1, 4, 6, 7, 11, 12

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023. 1, 5, 6, 7, 8, 9, 11, 12

[8] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Jeff Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. In *NeurIPS*, 2024. 11

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 12

[10] Zimu Liao, Siyan Chen, Rong Fu, Yi Wang, Zhongling Su, Hao Luo, Linning Xu, Bo Dai, Hengjie Li, Zhilin Pei, et al. Fisheye-gs: Lightweight and extensible gaussian splatting module for fisheye cameras. In *ECCV Workshop*, 2024. 1, 7, 8, 9

[11] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG*, 2019. 10

[12] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 5

[13] Keunhong Park, Philipp Henzler, Ben Mildenhall, Jonathan T Barron, and Ricardo Martin-Brualla. Camp: Camera preconditioning for neural radiance fields. *ACM TOG*, 2023. 5

[14] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1, 3, 6, 9

[15] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. 11

[16] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *CVPR*, 2024. 12