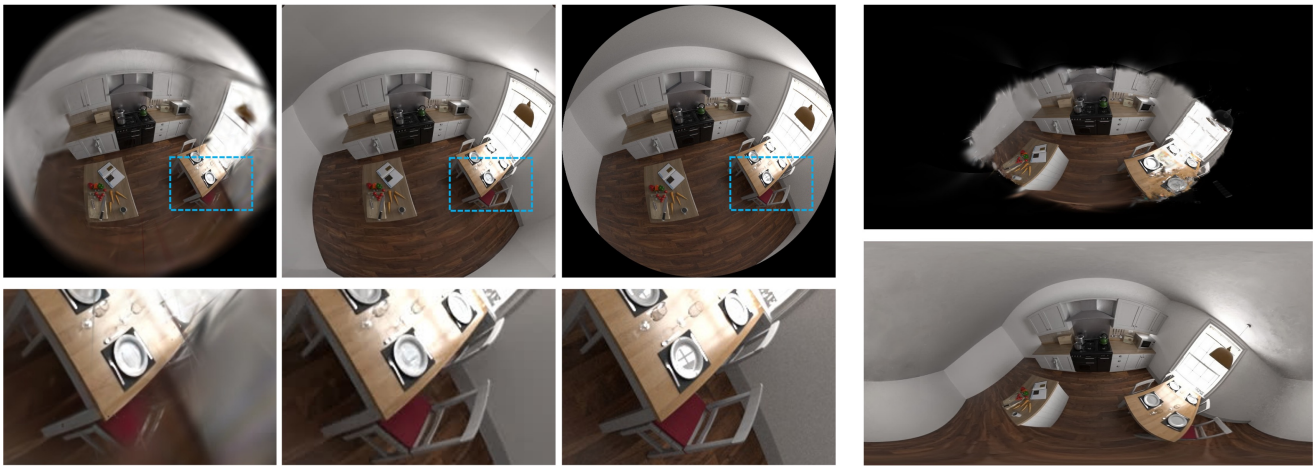
Self-Calibrating Gaussian Splatting for Large Field-of-View Reconstruction

Youming Deng[1] Wenqi Xian[2] Guandao Yang[3]
Leonidas Guibas[3] Gordon Wetzstein[3] Steve Marschner[1] Paul Debevec[2]
[1]Cornell University [2]Netflix Eyeline Studios [3]Stanford University



(a) Fisheye-GS [45] (b) Ours (c) Ground Truth (d) Rendered Panorama

Figure 1. We introduce Self-Calibrating Gaussian Splatting, a differentiable rasterization pipeline with a hybrid lens distortion field that can produce high-quality novel view synthesis results from uncalibrated wide-angle photographs. (a) Existing methods such as Fisheye-GS [45] fail to accurately handle complex lens distortions due to the traditional parametric distortion model. (b) Our method accurately models large distortions, especially in the peripheral regions, utilizing the entire highly distorted raw images for reconstruction. (d) Our method (bottom) provides extensive coverage, whereas conventional pipelines (top) can only recover the center.

Abstract

Large field-of-view (FOV) cameras can simplify and accelerate scene capture because they provide complete coverage with fewer views. However, existing reconstruction pipelines fail to take full advantage of large-FOV input data because they convert input views to perspective images, resulting in stretching that prevents the use of the full image. Additionally, they calibrate lenses using models that do not accurately fit real fisheye lenses in the periphery. We present a new reconstruction pipeline based on Gaussian Splatting that uses a flexible lens model and supports fields of view approaching 180 degrees. We represent lens distortion with a hybrid neural field based on an Invertible ResNet and use a cubemap to render wide-FOV images while retaining the efficiency of the Gaussian Splatting pipeline. Our system jointly optimizes lens distortion, camera intrinsics, camera poses, and scene representations using a loss measured directly against the original input pixels. We present extensive experiments on both synthetic and real-world scenes, demonstrating that our model accurately fits real-world fisheye lenses and that our end-to-end self-calibration approach provides higher-quality reconstructions than existing methods. More details and videos can be found at the project page: https://denghilbert.github.io/self-cali/.

1. Introduction

Large field-of-view (FOV) lenses, such as fisheye lenses, are widely used in robotics [19], virtual reality [39], and autonomous driving [18] because they capture scenes with fewer images, enabling efficient data acquisition for reconstruction and novel view synthesis (NVS)[46]. However, most modern 3D reconstruction systems based on Neural Radiance Fields (NeRFs)[48] and Gaussian Splatting (3DGS) [38] cannot be directly applied to this type of in-

put because they rely on perspective projection. As a result, fisheye images are typically re-projected into linear perspective images, and the resulting non-uniform sampling and stretching pose challenges for accurate camera calibration and scene reconstruction from wide-angle imagery.

Traditionally, imaging systems are pre-calibrated using specialized setups, such as calibration checkerboards. During calibration, polynomial distortion models [8, 56] are estimated and then used to re-project raw fisheye images into perspective ones for scene reconstruction. There are three important problems with this practice. First, resampling wide-field images leads to severe stretching and non-uniform sampling, as shown in Fig. 2a, where perspective images exhibit far less uniform sampling of directions than fisheyes. Second, traditional parametric distortion models lack sufficient expressiveness to accurately model large FOV (*e.g.,* 180°) lenses. Third, pre-calibration prevents lens parameters from being optimized end-to-end along with reconstruction, meaning the final reconstruction does not fully minimize the reconstruction loss.

To avoid the separate pre-calibration stage, recent works [36, 45, 49] integrate distortion modeling [13, 14, 52, 61] and optimization into NeRF and 3DGS. However, even when using photometric loss to refine distortion parameters alongside reconstruction, these approaches still exhibit significant misalignment in peripheral regions due to their reliance on traditional distortion models and single-plane projection. Both issues constrain the ability of current reconstruction methods to represent wide-FOV optical systems.

In this work, we introduce *Self-Calibrating Gaussian Splatting*, a differentiable rasterization pipeline that jointly optimizes lens distortion, camera intrinsics, camera poses, and scene representations using 3D Gaussians. Our approach effectively addresses the three aforementioned challenges, achieving high-quality reconstruction from large-FOV images without requiring resampling, pre-calibration, or polynomial distortion models.

To improve lens modeling, we replace the conventional parametric distortion model with a novel hybrid neural field that balances expressiveness and computational efficiency, as illustrated in Fig. 3. Our method uses invertible residual networks [6] to predict displacements on a normalized sparse grid, followed by bilinear interpolation to generate a continuous distortion field.

To mitigate the stretching caused by resampling to a single-plane perspective, we use cubemap sampling as in [15] (Fig. 2b), which significantly reduces stretching and yields a more uniform pixel density, even in peripheral areas. Gaussians project into each limited-FOV face of the cubemap with bounded distortion.

Finally, our method combines lens calibration with bundle adjustment and scene reconstruction into an integrated end-to-end self-calibration process that minimizes the rendering loss over all unknowns jointly, leading to lower reconstruction errors compared to pre-calibration of distortion and/or camera pose.

(a) Conventional Paradigm (b) Ours

Figure 2. **Conventional Paradigm vs. Our Method**. (a) Conventional approaches require reprojecting the image into perspective views compatible with 3DGS rasterization. As the field of view increases, pixel stretching becomes progressively severe, significantly compromising the quality of the reconstruction. (b) In contrast, our cubemap resampling strategy maintains a consistent pixel density across the entire field of view. This approach, combined with our hybrid distortion field, utilizes the peripheral regions (the annular area outside the blue box) without severe distortion or pixel stretching. Moreover, our method can handle fields of view up to 180°, as demonstrated by the green box, allowing for comprehensive and accurate reconstructions.

To validate our method, we conduct extensive experiments on both synthetic datasets and real-world scenes, including the FisheyeNeRF datasets [36], our own real-world captured scenes, and a synthetic dataset. Our system effectively calibrates extrinsics, intrinsics, and lens distortion, achieving better reconstruction performance compared to existing methods using uncalibrated fisheye cameras. Importantly, our system is not constrained to a single fisheye camera model; rather, it is designed to be flexible and adaptable, accommodating a wide range of cameras, from perspective to extreme fisheye, without requiring pre-calibration. This flexibility enables our method to fully leverage the unique capabilities of available lenses, ensuring comprehensive scene coverage and high-quality reconstructions.

2. Related Work

Camera Modeling and Lens Distortion. Lens distortion is an inherent property of all cameras. In general, nonlinear distortion can be formulated as:

$$\mathbf{x}_d = \mathbf{K} \cdot D\left(\pi\left(\mathbf{R} \cdot \mathbf{X} + \mathbf{t}\right)\right), \tag{1}$$

where $\mathbf{K}$ and $[\mathbf{R}|\mathbf{t}]$ represent the intrinsic and extrinsic parameters, respectively. $\mathbf{X}$ is a 3D point in world coordinates. $\pi(\cdot)$ denotes the pinhole projection, including dehomogenization to obtain 2D points $\mathbf{x}_n$ on the image plane. The

distortion model $D(r(\mathbf{x}_n))$ is parameterized as a polynomial function of the radial distance:

$$D(r(\mathbf{x}_n)) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots \quad (2)$$

where k_1, k_2, k_3, $\dots$ are the parameters of the Brown–Conrady model [9, 16], derived from calibration, and $r = \sqrt{x_n^2 + y_n^2}$. Early works [17, 24] propose to use panoramic lenses to model fisheye distortion but are limited by the spherical representation. Scaramuzza *et al.* [55] first proposed a unified model for large-FOV fisheye lenses, which has been adopted in several works[10, 41]. The most widely used fisheye model [34] describes distortion as a function of the angular distance from the projection center:

$$D(r(\mathbf{x}_n)) = \frac{\theta}{r}\left(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + \dots\right), \quad (3)$$

where $\theta = \arctan\left(\frac{r}{1}\right)$, and the distances of projected points to the image plane are normalized to 1. The 3D Gaussian Splatting method [38] assumes a standard perfect pinhole camera model and typically relies on COLMAP [56] to undistort images before reconstruction. To remove this constraint, recent methods [45, 49] adopt parametric models like Eq. (3) to extend 3D Gaussian Splatting techniques to fisheye images. However, these methods still depend heavily on camera calibration for accurate estimation and fix the projection in rasterization, limiting their generalizability to various camera types. Some works [49] introduce ray tracing into the rasterization pipeline and approximate Gaussian bounding using an icosahedron, which can potentially compromise rasterization efficiency. Other approaches [3, 33, 44] explore reconstruction from omnidirectional 360° panoramas, but the key difference is that panoramas require calibration to stitch two fisheye images together, which does not preserve raw geometric consistency at the stitching boundary. We address these limitations by introducing a hybrid distortion field that is compatible with the 3D Gaussian Splatting pipeline. Our experiments demonstrate that existing methods, such as Fisheye-GS [45] and ADOP [54], which incorporate traditional camera distortion models from Eqs. (2) and (3) into the rasterization process, are not expressive enough to handle the severe distortions present in large-FOV cameras.

Self-Calibrating Reconstruction. The bundle adjustment process can be extended to optimize camera lens parameters alongside poses, a process known as self-calibration [21, 31, 53, 62]. Camera calibration without a known calibration target is particularly challenging, as it relies on strong assumptions about scene structure and geometric priors to establish reliable correspondences [2, 5, 12]. Camera auto-calibration methods [21, 53, 62] extend this idea by deriving camera intrinsics from multi-view observations of unstructured scenes, an approach further advanced in recent studies [20, 22, 23, 29]. Several non-parametric models have been developed to ensure broad applicability across different camera and lens combinations [11, 26, 30, 43], while additional regularization is often required to maintain smooth underlying distortion [51]. With advances in differentiable rendering and rasterization pipelines [40, 60], recent works [36, 54, 58] have demonstrated that camera lens distortion can be optimized jointly with other parameters through a differentiable projection module. Prior works have also adapted NeRF for panoramic and fisheye-distorted inputs [28, 32, 42, 59]. These solutions typically rely on parametric models tailored for specific lenses, limiting their generalizability to a broader range of lens types. SCNeRF [36] models a residual projection matrix and residual raxel parameters [27], which are interpolated on a subsampled pixel grid. NeuroLens [58] optimizes lens parameters through an invertible neural network, while SC-OmniGS [33] optimizes camera parameters jointly with reconstruction but relies on calibrated captures. Building on insights from prior self-calibration methods, this work introduces a novel and efficient approach to modeling lens distortion, fully integrated with 3DGS [38].

3. Method

Given uncalibrated wide-angle captures, we aim to develop an algorithm that produces high-quality reconstructions using 3D Gaussians. Our method is designed to be robust against severe distortion in the peripheral regions of images and various wide-angle lens effects. First, we extend Gaussian Splatting to support a broader range of camera models, including fisheye lenses, as discussed in Sec. 3.2. To model lens distortion, we introduce a hybrid distortion field, enabling our approach to generalize across diverse real-world scenarios involving cameras with varying distortions. Second, we replace the traditional single-plane projection in 3DGS with a cubemap representation and introduce a distance-sorting strategy accordingly, as detailed in Sec. 3.3. Finally, we derive an efficient optimization for camera parameters, supporting self-calibration with distortion, as described in Sec. 3.4.

3.1. Background of Gaussian Splatting

A 3D Gaussian $G(x) = e^{-(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)}$ is parameterized by center $\mu_i \in \mathbb{R}^3$, covariance Σ_i, opacity σ_i, and color C_i, represented using spherical harmonics. The 3D Gaussians are projected [63] into $\mu_i^{2D} = \mathcal{P}(\mu_i, \Theta)$ and $\Sigma^{2D} = J_{\mathcal{P}}^T \Sigma J_{\mathcal{P}}$, where $J_{\mathcal{P}} \in \mathbb{R}^{3\times 2}$ is the affine approximation of the projection $\mathcal{P}$ at point μ_i, parameterized by Θ. For a pixel location u, the RGB color is produced with alpha blending [40, 60]:

$$\hat{I}(u) = \sum_{i=1}^{|G|} C_i \alpha_i \prod_{j \in \mathcal{N}_{<i}(G)} (1 - \alpha_j), \quad (4)$$

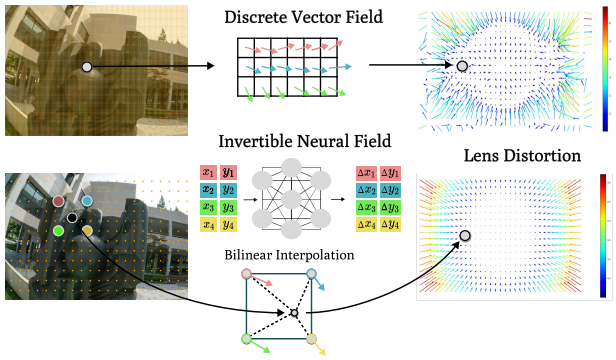


Figure 3. **Overview of Our Method**. In contrast to the explicit distortion vector field illustrated in the upper row, our hybrid approach maintains computational efficiency by leveraging explicit control points. Additionally, the regularization provided by the invertible neural field effectively balances the trade-off between the expressiveness and smoothness of the distortion field.

where $\mathcal{N}(G)$ is an ordered index of Gaussians sorted by depth, and α_i depends on σ_i, Σ_i^{2D}, and μ_i^{2D}. Finally, this set of 3D Gaussians is optimized using the L1 loss and D-SSIM [4] loss, along with adaptive control [38].

3.2. Lens Distortion Modeling

In this section, we extend the Gaussian Splatting technique to accommodate a broader class of camera lenses, including fisheye and wide-angle cameras, by modeling lens distortion. Lens distortion is typically captured by a distortion function defined in camera coordinates. A distortion function $\mathcal{D}_\theta : \mathbb{R}^2 \to \mathbb{R}^2$ parameterized by θ maps pixel locations from a rectified image to locations in a distorted image. Ideally, the mapping $\mathcal{D}_\theta$ should be: 1) expressive enough to model various lens distortions, 2) well-regularized so that it can be optimized together with the 3D scene, and 3) efficient, ensuring that it does not add significant computational overhead. While existing methods have explored using parametric camera models, grid-based methods, and deep-learning methods, none of these approaches perfectly satisfy all three criteria.

Grid-based method. The simplest way to implement a generic camera model is to explicitly optimize for the distortion in a grid of pixel coordinates and apply bilinear interpolation to extract a continuous distortion field:

$$\mathcal{D}_\theta(\mathbf{x}) = \mathbf{x} + \text{interp}(\mathbf{x}, \theta), \tag{5}$$

where the optimizable parameters $\theta \in \mathbb{R}^{H\times W\times 2}$ define an $H \times W$ grid storing 2D vectors representing the distortion. The bilinear interpolation function is given by $\text{interp}(\mathbf{x}, \theta) = W(\mathbf{x}, \theta) \cdot \theta$, where $W(\mathbf{x}, \theta) \in \mathbb{R}^{H\times W}$ represents the bilinear interpolation weights at location $\mathbf{x}$. Such a grid-based method is both expressive and efficient, as $W(\mathbf{x}, \theta)$ is sparse, and increasing the grid resolution allows for modeling more complex functions. However, the grid-based method lacks the smoothness required to model lens distortion properly, leading to overfitting and suboptimal solutions (Fig. 3 Top).

Invertible Residual Networks. An alternative way to model distortion is by using a neural network with an appropriate inductive bias. NeuroLens [58] proposes using an invertible ResNet [6] to represent non-linear lens distortions as a diffeomorphism. Specifically, the deformation mapping is modeled by a residual network:

$$\mathcal{D}_\theta(\mathbf{x}) = F_L \circ \cdots \circ F_1(\mathbf{x}), \quad F_i(z) = z + f_{\theta_i}^{(i)}(z), \tag{6}$$

where $f_{\theta_i}^{(i)}$ is a neural block parameterized by θ_i with a Lipschitz constant bounded by 1 (*i.e.*, $|f_\theta^{(i)}(x) - f_\theta^{(i)}(y)| < |x - y|$ for all x, y, and θ). $f_{\theta_i}^{(i)}$ represents a residual block with four linear layers. L denotes the total number of blocks, which is 5 in our case, and the circle denotes function composition. Such constraints make the network invertible, and its inverse can be obtained using a fixed-point algorithm [6]. In the supplementary, we also provide additional comparisons between iResNet and a regular ResNet.

While an invertible ResNet offers both expressiveness (*i.e.*, the ability to model various lenses) and regularization, it is computationally prohibitive to apply it directly to 3DGS. To backpropagate gradients to the alpha-blending weights α_i when rendering an image in Eq. (4), the computational graph for the backward passes of $\mathcal{D}_\theta$ must be maintained for each Gaussian. This is infeasible due to the large number of 3D Gaussians in a single scene, often reaching millions and leading to out-of-memory errors. This limitation motivates us to develop a more efficient solution that leverages the inherent inductive bias of the invertible ResNet while maintaining computational efficiency.

Hybrid Distortion Field. Given that the grid-based method is efficient yet tends to overfit, while the invertible ResNet has an appropriate inductive bias but is not efficient, we propose a hybrid method that combines the advantages of both. Specifically, we use the invertible ResNet to predict the flow field on a sparse grid and apply bilinear interpolation to each projected 2D Gaussian:

$$\mathcal{D}_\theta(\mathbf{x}) = \mathbf{x} + \text{interp}(\mathbf{x}, \mathcal{R}_\theta(\mathbf{P}_c) - \mathbf{P}_c), \tag{7}$$

where $\mathbf{P}_c \in \mathbb{R}^{H\times W\times 2}$ is a sparse grid of fixed control points (pixel locations, where $H \times W$ represents the resolution of control points rather than image resolution), and $\mathcal{R}_\theta$ is an invertible ResNet parameterized by θ.

Unlike existing hybrid neural fields [50], where networks are applied after grid interpolation, our approach uses iResNet to predict displacement vectors on a sparse grid, with bilinear interpolation applied to produce a continuous displacement field, as shown at the bottom of Fig. 3. The advantage of this architecture is that we only need to compute

the expensive forward and backward ResNet mappings for locations at $\mathbf{P}_c$, which scales with the grid resolution and is independent of the number of Gaussians in the scene. The additional operation required for each Gaussian is $\mathrm{interp}(\cdot)$, which is computationally affordable and parallelizable.

3.3. Cubemap for Large FOV

In order to apply our method to cameras with larger FOV, we extend the single-planar perspective projection to a cubemap projection [37, 57]. Mathematically, single-planar projection requires upsampling in the peripheral regions, and the sampling rate increases drastically as the FOV approaches 180°. In contrast, rendering with a cubemap maintains a relatively uniform pixel density from the image center to the edges, making it ideal for wide-angle rendering.

Single-Planar Projection. Given the parameters estimated from SfM [56], the parametric model is then applied to reproject the raw image into perspective images, as shown in the blue box in Fig. 2a. These reprojected images are then used for reconstruction through perspective-based pipelines like NeRF [48] or 3D Gaussian Splatting (3DGS) [38]. However, this process stretches the pixels in the peripheral regions, and the effect becomes more pronounced when the images are reprojected to larger FOV perspectives, as shown in the orange box. More specifically, the stretching rate of each pixel is defined by the inverse of Eq. (3), which exhibits a trend similar to $\tan(r)$, where r is the FOV angle from the center of the raw image. When the FOV of the reprojected image is 110° (as in the blue example), the upsampling rate from the blue circle to the box is approximately 1.4. However, when the FOV increases to 170°, as in the orange example, this rate increases to 11.4, inevitably sacrificing a significant amount of high-frequency information for reconstruction.

Moreover, to preserve central details, the resolution of undistorted images needs to be higher, as the pixel density at the center of the raw image should ideally match that of perspective ones. For example, when undistorting a fisheye image in Fig. 2a, the resulting perspective image in the orange region would have an extremely high resolution, making it computationally expensive to render. A common solution is to crop away the periphery, following COLMAP's solution [56], but this strategy contradicts our intention of using a fisheye camera to capture wide-angle information.

Multi-Planar Projections. Inspired by the representation of environment maps using cubemaps [25] and hemicube [15] in computer graphics, we propose representing extreme wide-angle renderings with cubemap projections, each covering 90° FOV and oriented orthogonally to one another, as illustrated in Fig. 2b. By resampling across the cubemap faces, we can render perspective or distorted images with FOVs even larger than 180°. Fast rasterization is first applied to obtain each face of the cubemap. For each rendered pixel, we look up its corresponding position in the constructed cubemap. Our hybrid distortion field then resamples from the lookup table to achieve the distorted rendering. In practice, it is not necessary to render all faces of the cubemap at once, as the number of cubemap faces may vary for different FOV camera lenses.

The resampling step involves only a simple coordinate transformation along with our hybrid field distortion. The distance from the shared camera center to each plane is normalized to 1. To render a pixel outside the 90° FOV at $(x, y, -1)$ where $x > 1$ and $|y| < 1$, for instance, the pixel on the right face can be obtained as $(\frac{-1}{x}, \frac{y}{x}, -1)$ in camera coordinates, looking toward the right side. When considering lens distortion, the sampling mapping is distorted according to Eq. (7), altering the lookup on the right face to $(\frac{-1}{x'}, \frac{y'}{x'}, -1)$, where $(x', y') = \mathcal{D}_\theta(x, y)$. By doing so, the entire distortion field can be directly applied to the cubemap for large FOV rendering. The entire resampling process is fully differentiable, making it directly applicable in our hybrid distortion field as a plug-and-play module.

Gaussian Sorting. 3DGS [38] constructs an ordered set $\mathcal{N}(G)$ of Gaussians before alpha blending [38]. Gaussians are sorted based on their orthogonal projection distance to the image plane. This approach is valid as long as a single Gaussian is not projected onto multiple faces. However, with the cubemap representation, Gaussians can span the boundary between two faces, leading to multiple projections with inconsistent ordering across faces. This discrepancy introduces intensity discontinuities at the boundaries. To address this issue, we replace the original sorting strategy with a distance-based approach, ordering Gaussians by their distance from the camera center. This ensures that the rasterization order remains consistent across all cubemap faces, thereby alleviating intensity discontinuities. Due to the affine approximation used in [38, 63], the 2D covariance of Gaussians near cubemap face boundaries still has a slight influence on the final rendering, which we further discuss in Sec. 5 and the supplementary material.

3.4. Optimization of Camera Parameters

Our pipeline differentiates all camera parameters. We theoretically derive the gradient for all camera parameters, including extrinsics and intrinsics, making the camera module completely differentiable and capable of being optimized alongside distortion modeling. All gradient calculations are implemented with a native CUDA kernel. A comprehensive mathematical derivation and experimental results are provided in the supplementary.

4. Experiments

In this section, we first briefly introduce our data preparation pipeline in Sec. 4.1. We compare our method against various baselines for scene reconstruction with large-FOV

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-perspective [38]	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
3DGS-COLMAP [38]	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
Adop-GS [54]	0.829	22.59	0.200	0.755	22.12	0.289	0.646	19.96	0.314	0.758	21.35	0.294	0.741	21.37	0.306	0.726	22.48	0.254
Fisheye-GS [45]	0.785	21.68	0.110	0.754	23.29	0.166	0.615	20.23	0.214	0.728	22.11	0.160	0.722	21.37	0.218	0.697	22.38	0.177
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 1. **Quantitative Evaluation on the FisheyeNeRF Dataset** [36]. We evaluate our method on a challenging real-world benchmark. Our method consistently outperforms existing baselines. Additional qualitative results are shown in Fig. 4.

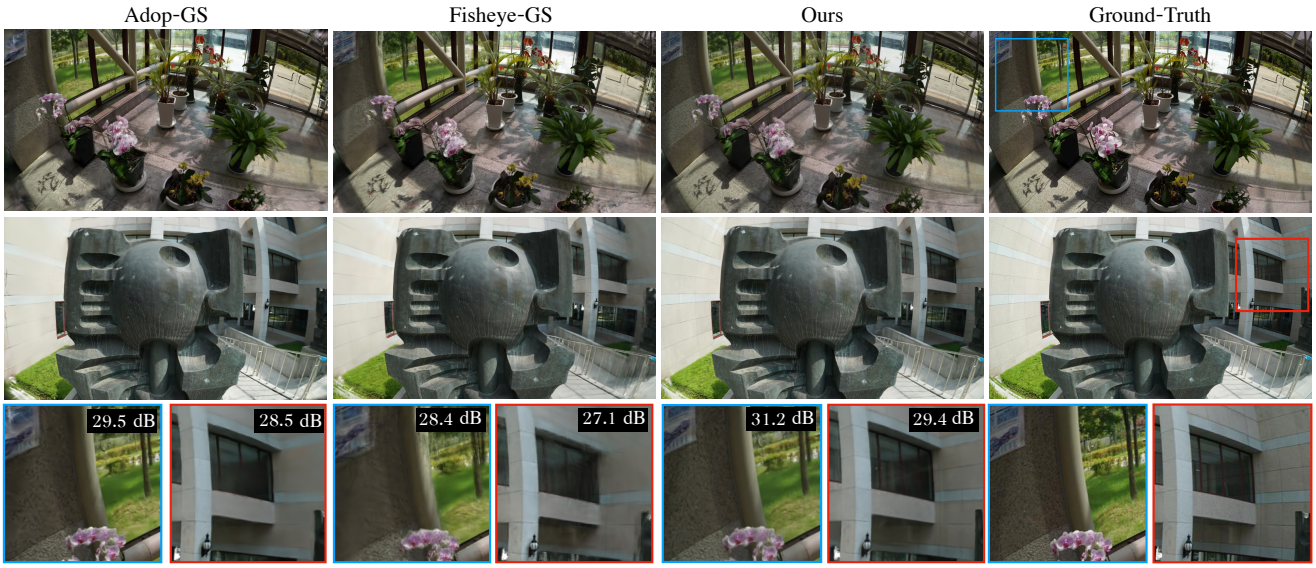


Figure 4. **Qualitative Comparisons with Baselines on the FisheyeNeRF Dataset** [36]. The images show comparisons across different scenes using two baselines (*e.g.,* ADOP-GS [54] and Fisheye-GS [45]) and our method. PSNRs are computed for each patch.

input (Sec. 4.2 and Sec. 4.3), followed by ablation studies for both the neural distortion model and the cubemap projection in Sec. 4.4.

4.1. Data Acquisition

We customized a camera module in the Mitsuba ray tracer [35] to incorporate fisheye camera parameters derived from the open-source Lensfun database [1]. Using a 180° fisheye camera, we rendered three scenes from [7]. We generated a training set with both perspective and fisheye views for baselines and our method, respectively. Additionally, we captured several real-world datasets using different uncalibrated cameras to evaluate our method. These datasets consist of casual walk-around video footage captured using a camera with an approximate 150° FOV. Furthermore, we tested our method on the existing FisheyeNeRF benchmark dataset [36], which contains fisheye images with an FOV of approximately 120°.

4.2. Comparisons to Traditional Lens Models

We first validate that our hybrid field representation models large-FOV cameras better than traditional polynomial distortion models [8, 56]. Using the FisheyeNeRF dataset [36], we compare our method against four baselines. The first baseline is Vanilla 3DGS [38], which only uses a perspective camera model. Second, we directly apply the polynomial distortion model estimated by COLMAP [56] to 2D Gaussians after the rasterization of 3DGS [38]. Third, we evaluate Fisheye-GS [45], which modifies the projection of Gaussians to a specific fisheye parametric model. Finally, we re-implemented ADOP [54] using an omnidirectional camera model as the fourth baseline.

We report quantitative results in Tab. 1, and our method consistently outperforms all baselines. We also show novel-viewpoint renderings in Fig. 4. Baseline methods such as ADOP [54] and Fisheye-GS [45] tend to produce more artifacts in the corners of the test views, whereas our method reconstructs finer details.

To evaluate how our method performs with increasing FOV, we conducted an additional experiment on two datasets with different camera FOVs: real-world scenes captured with 150° cameras and a synthetic dataset with a 180° FOV. As the FOV approaches 180°, as shown in Fig. 5, our method successfully handles peripheral regions, whereas Fisheye-GS [45] fails, producing spiky Gaussians in the surroundings and introducing artifacts at

(a) Fisheye-GS (b) Ours (c) GT (d) Fisheye-GS (e) Ours (f) GT

Figure 5. **Qualitative Comparisons with Fisheye-GS [45]**. To validate our hybrid distortion modeling, we further compare our method with Fisheye-GS [45] on larger FOV scenes, including real-world captures using 150° cameras (a–c) and simulations using a 180° camera (d–f) in Mitsuba [35]. Our method successfully recovers details in peripheral regions, whereas Fisheye-GS [45] struggles.

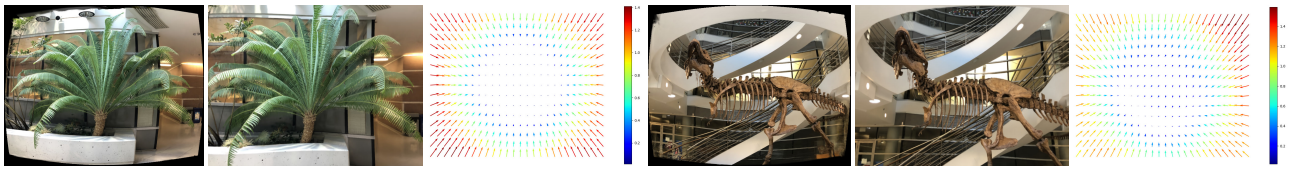


Figure 6. **Qualitative Results of Different Distortions**. We show renderings in both distorted and perspective views. We also visualize the distortion field after optimization. Notably, our method can figure out a combination of radial and tangential distortions for T-Rex.

the center that blur the rendering. The quantitative evaluation of Fig. 5 can be found in the supplementary material.

Our method can also be applied to different types of lens distortion. We introduce radial and tangential distortions to images in the LLFF dataset [47], using camera parameters derived from the Lensfun database [1]. By applying these parameters, we generate a combination of distortions, as demonstrated in the T-Rex scene. We optimize our model using these distorted images and obtain the distortion map learned by our lens model. Fig. 6 shows that our method accurately recovers various types of camera distortion without manual calibration or access to the physical camera.

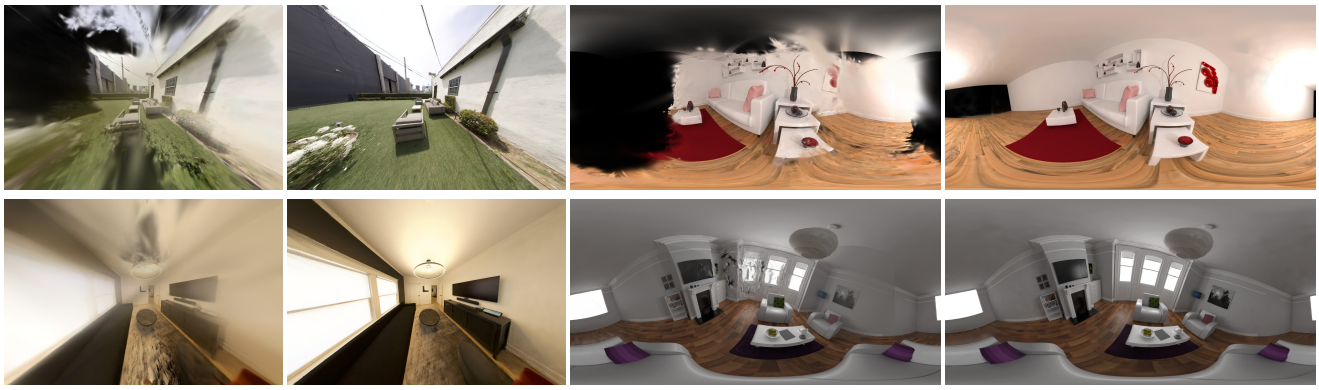
4.3. Large FOV Reconstruction with Few Captures

By enabling the use of large-FOV views, our method can reconstruct scenes with fewer images than would be needed for narrower perspective views. We evaluate our method on 150° real-world captures and 180° synthetic scenes. For real-world scenes, we follow the conventional paradigm of using COLMAP to first resample the images to perspective—including peripheral cropping performed by COLMAP [56]—before reconstruction. For synthetic scenes, we can flexibly modify the camera model and render images with a regular 90° FOV as input for 3DGS [38]. Since our method supports perspective rendering after training, we compare it against the baseline using a set of perspective hold-out cameras for a fair evaluation. Quantitative evaluation is conducted only on synthetic data, as obtaining ground truth perspective images for real-world captures is not feasible.

Method	Num	SSIM	PSNR	LPIPS
3DGS [38]	200	0.654	19.08	0.332
Ours	100	**0.800**	**29.01**	**0.231**
	50	0.735	26.26	0.267
	25	0.709	24.59	0.292
	10	0.615	22.77	0.356

Table 2. **Evaluation on Mitsuba Scenes**. We compare our method with Vanilla 3DGS [38] to demonstrate the advantages of using wide-angle cameras over regular-FOV cameras. The testing views are rendered in perspective. Our method achieves better performance and coverage, even with fewer captures.

We report quantitative results in Tab. 2, where our method outperforms the baseline even with far fewer input views, down to 10-15%. The trajectories of Room 1 and Garden are similar, both focusing inward, while Room 2 and Studio involve a walk-around motion covering the entire space. As a result, we observe large incomplete regions in the first row of Fig. 7 for the baseline, whereas our method achieves complete reconstruction. The walk-around scenes shown in the last row of Fig. 7 exhibit noticeable artifacts and floating elements, primarily due to small-FOV captures or cropping in COLMAP. These experiments demonstrate the efficiency of our approach compared to



(a) 3DGS w/ COLMAP (b) Ours (c) 3DGS w/ Regular Camera (d) Ours

Figure 7. **Qualitative Comparison of Reconstruction Coverage**. We evaluate our method on real-world scenes where (a) 3DGS [38] relies on an SfM method like COLMAP [56], which crops the highly distorted periphery when the FOV is large. (b) Our method recovers a wider region with fewer artifacts at the center. We also visualize reconstructions using panorama views of both (c) 3DGS [38] and (d) our method. 3DGS [38] only supports perspective inputs captured with regular-FOV cameras, whereas our method can be directly applied to 180° inputs, demonstrating better coverage even with fewer training views, as shown in Tab. 2.

Explicit Grid	iResNet	PSNR	SSIM	LPIPS
✗	✓	Out-of-Memory		
✗	✗	14.19	0.421	0.561
✓	✗	19.79	0.569	0.309
✓	✓	**23.67**	**0.777**	**0.151**

Table 3. **Ablations of Hybrid Field.** We conduct ablation studies on our full method, including an explicit grid initialized with COLMAP's traditional polynomial distortion parameters, to demonstrate the necessity of adopting a hybrid neural distortion field representation in FisheyeNeRF [36].

Projection	PSNR	SSIM	LPIPS
Single Plane	24.10	0.676	0.312
Cubemap	**29.01**	**0.792**	**0.253**

Table 4. **Ablation of Cubemap**. We evaluate single-plane and cubemap projections using the same hybrid field on Mitsuba scenes, where the FOV is close to 180°, causing significant degradation in the single-plane projection.

regular-FOV cameras, achieving high-quality reconstruction with significantly fewer captures, as shown in Tab. 2.

4.4. Ablation Studies

Hybrid Field. To verify the effectiveness of each module in our hybrid field, we isolate individual components (either iResNet or the explicit grid) and evaluate their performance separately. We report quantitative results in Tab. 3 on FisheyeNeRF [36] scenes, showing that neither module alone achieves optimal performance compared to our full method. To test the grid alone, we use an explicit grid of learnable displacement vectors. This strategy provides only a slight improvement, primarily because the optimization of the explicit grid is prone to overfitting and can become trapped in local minima. We visualize the difference between the explicit grid and our hybrid field distortion flow in Fig. 3. We cannot report results for iResNet without the grid, as explained in the last paragraph of **Invertible Residual Networks** in Sec. 3.2. Our full method (iResNet + control grid) achieves the best performance by combining the expressiveness of iResNet with the computational efficiency of the explicit grid. This ablation verifies the necessity of our hybrid representation for modeling distortion.

Cubemap Resampling. To validate the necessity of cubemap resampling for wide-FOV reconstruction, we compare results obtained using our hybrid distortion field with single-plane projection versus cubemap resampling. We evaluate the performance of these two methods in the Mitsuba synthetic scenes. As presented in Tab. 4, using cubemap projections produces significantly better results when reconstructing images captured by 180° fisheye lenses. Additional qualitative comparisons between single-plane and cubemap projections are provided in the supplementary.

5. Discussions

This paper presents a method for optimizing 3D Gaussian representations while self-calibrating camera parameters and lens distortion. Our approach enables the use of large field-of-view captures to achieve efficient and high-quality reconstruction without cumbersome pre-calibration. Even with fewer input captures, our method maintains comprehensive scene coverage and reconstruction quality.

Acknowledgement

This work was supported in part by the National Science Foundation under grant 2212084 and the Vannevar Bush Faculty Fellowship. We want to express gratitude to Xichen Pan, Xiangzhi Tong, Junyi Zhang, Gene Chou, Julien Philip, Li Ma, Hansheng Chen, Jan Ackermann, and Eric Chen for their discussion and suggestions on this paper.

References

[1] Lensfun. https://lensfun.github.io/. 6, 7

[2] Miguel Alemán-Flores, Luis Alvarez, Luis Gomez, and Daniel Santana-Cedrés. Automatic lens distortion correction using one-parameter division models. *Image Processing On Line*, 2014. 3

[3] Jiayang Bai, Letian Huang, Jie Guo, Wen Gong, Yuanqi Li, and Yanwen Guo. 360-gs: Layout-guided panoramic gaussian splatting for indoor roaming. *arXiv preprint arXiv:2402.00763*, 2024. 3

[4] Allison H Baker, Alexander Pinard, and Dorit M Hammerling. Dssim: a structural similarity index for floating-point data. *arXiv preprint arXiv:2202.02616*, 2022. 4

[5] João Pedro Barreto and Helder Araujo. Geometric properties of central catadioptric line images and their application in calibration. *IEEE TPAMI*, 2005. 3

[6] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *ICML*, 2019. 2, 4

[7] Benedikt Bitterli. Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 6

[8] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 2, 6

[9] Duane Brown. Decentering distortion of lenses. *Photogrammetric engineering*, 1996. 3

[10] Martin Bujnak, Zuzana Kukelova, and Tomas Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *ACCV*, 2010. 3

[11] Federico Camposeco, Torsten Sattler, and Marc Pollefeys. Non-parametric structure-based calibration of radially symmetric cameras. In *ICCV*, 2015. 3

[12] Robert Carroll, Maneesh Agrawala, and Aseem Agarwala. Optimizing content-preserving projections for wide-angle images. *ACM TOG*, 2009. 3

[13] Manmohan Chandraker, Sameer Agarwal, Fredrik Kahl, David Nistér, and David Kriegman. Autocalibration via rank-constrained estimation of the absolute quadric. In *CVPR*, 2007. 2

[14] Manmohan Chandraker, Sameer Agarwal, David Kriegman, and Serge Belongie. Globally optimal algorithms for stratified autocalibration. *IJCV*, 2010. 2

[15] Michael F. Cohen and Donald P. Greenberg. The hemicube: a radiosity solution for complex environments. In *SIGGRAPH*, 1985. 2, 5

[16] Alexander Eugen Conrady. Decentred lens-systems. *Monthly notices of the royal astronomical society*, 1919. 3

[17] Jonathan Courbon, Youcef Mezouar, Laurent Eckt, and Philippe Martinet. A generic fisheye camera model for robotic applications. In *IROS*, 2007. 3

[18] Zhaopeng Cui, Lionel Heng, Ye Chuan Yeo, Andreas Geiger, Marc Pollefeys, and Torsten Sattler. Real-time dense mapping for self-driving vehicles using fisheye cameras. In *ICRA*, 2019. 1

[19] Varuna De Silva, Jamie Roche, and Ahmet Kondoz. Robust fusion of lidar and wide-angle camera data for autonomous mobile robots. *Sensors*, 2018. 1

[20] Youming Deng, Xueting Li, Sifei Liu, and Ming-Hsuan Yang. Physics-based indirect illumination for inverse rendering. In *3DV*, 2024. 3

[21] Frederic Devernay and Olivier Faugeras. Straight lines have to be straight. *Machine vision and applications*, 2001. 3

[22] Jakob Engel, Vladyslav Usenko, and Daniel Cremers. A photometrically calibrated benchmark for monocular visual odometry. *arXiv preprint arXiv:1607.02555*, 2016. 3

[23] Jiading Fang, Igor Vasiljevic, Vitor Guizilini, Rares Ambrus, Greg Shakhnarovich, Adrien Gaidon, and Matthew R Walter. Self-supervised camera self-calibration from video. In *ICRA*, 2022. 3

[24] Christopher Geyer and Kostas Daniilidis. A unifying theory for central panoramic systems and practical implications. In *ECCV*, 2000. 3

[25] Ned Greene. Environment mapping and other applications of world projections. *IEEE computer graphics and Applications*, 1986. 5

[26] Michael D Grossberg and Shree K Nayar. A general imaging model and a method for finding its parameters. In *ICCV*, 2001. 3

[27] Michael D Grossberg and Shree K Nayar. The raxel imaging model and ray-based calibration. *IJCV*, 2005. 3

[28] Kai Gu, Thomas Maugey, Sebastian Knorr, and Christine Guillemot. Omni-nerf: neural radiance field from 360 image captures. In *ICME*, 2022. 3

[29] Hyowon Ha, Sunghoon Im, Jaesik Park, Hae-Gon Jeon, and In So Kweon. High-quality depth from uncalibrated small motion clip. In *CVPR*, 2016. 3

[30] Richard Hartley and Sing Bing Kang. Parameter-free radial distortion correction with center of distortion estimation. *IEEE TPAMI*, 2007. 3

[31] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3

[32] Huajian Huang, Yingshu Chen, Tianjian Zhang, and Sai-Kit Yeung. 360roam: Real-time indoor roaming using geometry-aware 360 radiance fields. *SIGGRAPH Asia*, 2022. 3

[33] Huajian Huang, Yingshu Chen, Longwei Li, Hui Cheng, Tristan Braud, Yajie Zhao, and Sai-Kit Yeung. Sc-omnigs: Self-calibrating omnidirectional gaussian splatting. *arXiv preprint arXiv:2502.04734*, 2025. 3

[34] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015. 3

[35] 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. 6, 7

[36] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *ICCV*, 2021. 2, 3, 6, 8

[37] Hao Jiang, Gangyi Jiang, Mei Yu, Yun Zhang, You Yang, Zongju Peng, Fen Chen, and Qingbo Zhang. Cubemap-based perception-driven blind quality assessment for 360-degree images. *IEEE TIP*, 2021. 5

[38] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023. 1, 3, 4, 5, 6, 7, 8

[39] Gregor Klančar, Matej Kristan, and Rihard Karba. Wide-angle camera distortions and non-uniform illumination in mobile robot tracking. *Robotics and Autonomous Systems*, 2004. 1

[40] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, 2021. 3

[41] Zuzana Kukelova, Jan Heller, Martin Bujnak, Andrew Fitzgibbon, and Tomas Pajdla. Efficient solution to the epipolar geometry for radially distorted cameras. In *ICCV*, 2015. 3

[42] Shreyas Kulkarni, Peng Yin, and Sebastian Scherer. 360fusionnerf: Panoramic neural radiance fields with joint guidance. In *IROS*, 2023. 3

[43] Hongdong Li and Richard Hartley. Plane-based calibration and auto-calibration of a fish-eye camera. In *ACCV*, 2006. 3

[44] Longwei Li, Huajian Huang, Sai-Kit Yeung, and Hui Cheng. Omnigs: Omnidirectional gaussian splatting for fast radiance field reconstruction using omnidirectional images. *arXiv preprint arXiv:2404.03202*, 2024. 3

[45] 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, 2, 3, 6, 7

[46] Chuiwen Ma, Liang Shi, Hanlu Huang, and Mengyuan Yan. 3d reconstruction from full-view fisheye camera. *arXiv preprint arXiv:1506.06273*, 2015. 1

[47] 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. 7

[48] 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

[49] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *SIGGRAPH ASIA*, 2024. 2, 3

[50] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 4

[51] Linfei Pan, Marc Pollefeys, and Viktor Larsson. Camera pose estimation using implicit distortion models. In *CVPR*, 2022. 3

[52] Marc Pollefeys and Luc Van Gool. Stratified self-calibration with the modulus constraint. *TPAMI*, 1999. 2

[53] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *IJCV*, 1999. 3

[54] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM TOG*, 2022. 3, 6

[55] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *IROS*, 2006. 3

[56] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 2, 3, 5, 6, 7, 8

[57] Liang Wan, Tien-Tsin Wong, and Chi-Sing Leung. Isocube: Exploiting the cubemap hardware. *TVCG*, 2007. 5

[58] Wenqi Xian, Aljaž Božič, Noah Snavely, and Christoph Lassner. Neural lens modeling. In *CVPR*, 2023. 3, 4

[59] Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, et al. Vr-nerf: High-fidelity virtualized walkable spaces. In *SIGGRAPH Asia*, 2023. 3

[60] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM TOG*, 2019. 3

[61] Cyril Zeller and Olivier Faugeras. *Camera self-calibration from video sequences: the Kruppa equations revisited*. PhD thesis, INRIA, 1996. 2

[62] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, 1999. 3

[63] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Visualization*, 2001. 3, 5