Supplementary Material for OmniGS: Fast Radiance Field Reconstruction using Omnidirectional Gaussian Splatting

Longwei Li[1] Huajian Huang[2] Sai-Kit Yeung[2] Hui Cheng[1]
[1]Sun Yat-sen University [2] The Hong Kong University of Science and Technology
lilw23@mail2.sysu.edu.cn, hhuangbg@connect.ust.hk, saikit@ust.hk, chengh9@mail.sysu.edu.cn

A. Detailed Backward Gradient Derivation

A.1. Prerequisites

First, let us reconsider the α-blending model in 3D Gaussian Splatting (3DGS) [6], which is used for rendering the final color of each pixel on the output image:

$$C = \sum_{i=1}^{N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{1}$$

where N is the number of 3D Gaussians near enough to this pixel, i.e. the projected N Gaussian centers are within a certain distance threshold from the center of this pixel. For perspective cameras, these N Gaussians are sorted by their t_z (local depth), from nearest to farthest. But for omnidirectional cameras, they are sorted by t_r (local distance to the camera center), from nearest to farthest.

For the i-th 3D Gaussian, color c_i is determined by the relative position from the camera center to its Gaussian mean $\mathbf{m}$ (world coordinate), and its Spherical Harmonics coefficients. Neither of them is affected by the camera model, so there is no need to change this part of the gradient.

The sampled intensity α_i is determined by opacity o_i and the sampled value on its 2D Gaussian distribution:

$$\alpha_i = o_i G_i(\Delta \mathbf{p}_i), \tag{2}$$

where $\Delta \mathbf{p}_i = \mathbf{p}_i - \mathbf{p}_s$ is the difference vector between projected Gaussian center $\mathbf{p}_i$ and sampling pixel position $\mathbf{p}_s$. The scope of o_i stops here, so it is also not influenced by the camera model, and we don't need to change its gradient.

The sampling on the 2D Gaussian function $G_i(\cdot)$ is:

$$G_i(\Delta \mathbf{p}_i) = \exp\left(-\frac{1}{2} (\Delta \mathbf{p}_i)^{\mathrm{T}} \tilde{\mathbf{\Sigma}}_i^{-1} (\Delta \mathbf{p}_i)\right). \tag{3}$$

where $\mathbf{p}_s$ is a constant from the view of sampled Gaussians. So it is clear that what we need to determine before sampling each point is actually the 2D Gaussian center $\mathbf{p}_i$ and inverse covariance $\tilde{\mathbf{\Sigma}}_i^{-1}$. Without loss of generality, let us consider a certain Gaussian and elide the subscript i. Until now, we can derive the related gradient w.r.t. loss $\mathcal{L}$ as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{m}} = \sum_{k=1}^{M} \left[\frac{\partial \mathcal{L}}{\partial c} \frac{\partial c}{\partial \mathbf{m}} + \frac{\partial \mathcal{L}}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial G_k} \left(\frac{\partial G_k}{\partial \tilde{\mathbf{\Sigma}}^{-1}} \frac{\partial \tilde{\mathbf{\Sigma}}^{-1}}{\partial \tilde{\mathbf{\Sigma}}} \frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{m}} + \frac{\partial G_k}{\partial \mathbf{p}} \frac{\partial \mathbf{p}}{\partial \mathbf{m}} \right) \right], \tag{4}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \sum_{k=1}^{M} \left(\frac{\partial \mathcal{L}}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial G_k} \frac{\partial G_k}{\partial \tilde{\mathbf{\Sigma}}^{-1}} \frac{\partial \tilde{\mathbf{\Sigma}}^{-1}}{\partial \tilde{\mathbf{\Sigma}}} \frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{q}} \right), \tag{5}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{S}} = \sum_{k=1}^{M} \left(\frac{\partial \mathcal{L}}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial G_k} \frac{\partial G_k}{\partial \tilde{\mathbf{\Sigma}}^{-1}} \frac{\partial \tilde{\mathbf{\Sigma}}^{-1}}{\partial \tilde{\mathbf{\Sigma}}} \frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{S}} \right), \tag{6}$$

where $\mathbf{m}$ is the world-space Gaussian mean position, $\mathbf{q}$ is its rotation quaternion, and M is the total number of instances generated by this Gaussian in all rendering tiles. The Gaussian covariance is determined by $\mathbf{q}$ and scale $\mathbf{S}$.

As stated before, the color branch $\frac{\partial \mathcal{L}}{\partial c} \frac{\partial c}{\partial \mathbf{m}}$ does not change with camera model. For the intensity branch, $\frac{\partial \mathcal{L}}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial G_k}$, $\frac{\partial G_k}{\partial \tilde{\mathbf{\Sigma}}^{-1}} \frac{\partial \tilde{\mathbf{\Sigma}}^{-1}}{\partial \tilde{\mathbf{\Sigma}}}$ and $\frac{\partial G_k}{\partial \mathbf{p}}$ are relations between final image loss and the projected 2D Gaussians. They have already been given by [6] (some manually derived, the others given by PyTorch autograd). We keep them the same.

Let $\mathbf{\Sigma}$ denote the 3D Gaussian covariance in the world coordinate system, $\mathbf{R}$ denote the Gaussian rotation matrix converted from quaternion $\mathbf{q}$. We have:

$$\frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{q}} = \frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{\Sigma}} \frac{\partial \mathbf{\Sigma}}{\partial \mathbf{R}} \frac{\partial \mathbf{R}}{\partial \mathbf{q}}, \tag{7}$$

$$\frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{S}} = \frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{\Sigma}} \frac{\partial \mathbf{\Sigma}}{\partial \mathbf{S}}, \tag{8}$$

where $\frac{\partial \mathbf{\Sigma}}{\partial \mathbf{R}} \frac{\partial \mathbf{R}}{\partial \mathbf{q}}$ and $\frac{\partial \mathbf{\Sigma}}{\partial \mathbf{S}}$ are completely in the 3D world coordinate system so have nothing to do with the camera model. They have also been implemented by [6].

Recall that we use the local affine approximation method [10] to perform the projection:

$$\tilde{\mathbf{\Sigma}} \approx \mathbf{J}\mathbf{W}\mathbf{\Sigma}\mathbf{W}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}, \tag{9}$$

where $\mathbf{W}$ is the rotation part of the 4×4 transformation matrix $\mathbf{T}_{\mathrm{cw}}$ from the world coordinate system to the camera space. This transformation (also known as the camera pose) is calibrated by a sparse SfM algorithm (we use the openMVG [8] implementation). It constructs a relationship between the world position $\mathbf{m} = [m_x, m_y, m_z]^{\mathrm{T}}$ and the local position (camera coordinate system) $\mathbf{t} = [t_x, t_y, t_z]^{\mathrm{T}}$ of the mean of a Gaussian:

$$\mathbf{t} = \mathbf{T}_{\mathrm{cw}} * \mathbf{m} = \mathbf{W}\mathbf{m} + \mathbf{t}_{\mathrm{cw}}. \tag{10}$$

where $\mathbf{t}_{\mathrm{cw}}$ is the translation part of camera pose. We denote:

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{00} & \mathbf{W}_{01} & \mathbf{W}_{02} \\ \mathbf{W}_{10} & \mathbf{W}_{11} & \mathbf{W}_{12} \\ \mathbf{W}_{20} & \mathbf{W}_{21} & \mathbf{W}_{22} \end{bmatrix}. \tag{11}$$

$\mathbf{J}$ is the Jacobian of the camera projection:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{00} & \mathbf{J}_{01} & \mathbf{J}_{02} \\ \mathbf{J}_{10} & \mathbf{J}_{11} & \mathbf{J}_{12} \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{\partial p_x}{\partial t_x} & \frac{\partial p_x}{\partial t_y} & \frac{\partial p_x}{\partial t_z} \\ \frac{\partial p_y}{\partial t_x} & \frac{\partial p_y}{\partial t_y} & \frac{\partial p_y}{\partial t_y} \\ 0 & 0 & 0 \end{bmatrix}, \tag{12}$$

Because 3D world-coordinate covariance $\mathbf{\Sigma}$ has nothing to do with $\mathbf{m}$, and the camera rotation matrix $\mathbf{W}$ is regarded as a constant here, we can derive from Eq. (9) that:

$$\frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{m}} = \frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{J}} \frac{\partial \mathbf{J}}{\partial \mathbf{t}} \frac{\partial \mathbf{t}}{\partial \mathbf{m}} \tag{13}$$

To derive the items within $\mathbf{J}$, recall the equirectangular camera model we used to project the local $\mathbf{t}$:

$$\begin{bmatrix} lon \\ lat \end{bmatrix} = \begin{bmatrix} \arctan2(t_x/t_z) \\ \arcsin(t_y/t_r) \end{bmatrix}, \tag{14}$$

$$\begin{bmatrix} s_x \\ s_y \end{bmatrix} = \begin{bmatrix} lon/\pi \\ 2lat/\pi \end{bmatrix}, \tag{15}$$

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} (s_x+1)W/2 \\ (s_y+1)H/2 \end{bmatrix}, \tag{16}$$

$[s_x, s_y]^{\mathrm{T}}$ and $[p_x, p_y]^{\mathrm{T}}$ are the screen-space and image pixel coordinates respectively, $\arctan2$ is the 4-quadrant inverse tangent. Here we derive the gradient of the camera model:

$$\mathbf{J}_{00} = \frac{\partial p_x}{\partial t_x} = +\frac{W}{2\pi} \cdot \frac{t_z}{t_x^2 + t_z^2}, \tag{17}$$

$$\mathbf{J}_{01} = \frac{\partial p_x}{\partial t_y} = 0, \tag{18}$$

$$\mathbf{J}_{02} = \frac{\partial p_x}{\partial t_z} = -\frac{W}{2\pi} \cdot \frac{t_x}{t_x^2 + t_z^2}, \tag{19}$$

$$\mathbf{J}_{10} = \frac{\partial p_y}{\partial t_x} = -\frac{H}{\pi} \cdot \frac{t_x t_y}{t_r^2\sqrt{t_x^2 + t_z^2}}, \tag{20}$$

$$\mathbf{J}_{11} = \frac{\partial p_y}{\partial t_y} = +\frac{H}{\pi} \cdot \frac{\sqrt{t_x^2 + t_z^2}}{t_r^2}, \tag{21}$$

$$\mathbf{J}_{12} = \frac{\partial p_y}{\partial t_z} = -\frac{H}{\pi} \cdot \frac{t_z t_y}{t_r^2\sqrt{t_x^2 + t_z^2}}, \tag{22}$$

where $t_r = \sqrt{t_x^2 + t_y^2 + t_z^2}$, W,H are image width, height.

Till now, we have collected the prerequisites for derivation. What we need to do is propagating the gradient to $\frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{m}}$ (for $\mathbf{m}$, covariance branch), $\frac{\partial \mathbf{p}}{\partial \mathbf{m}}$ (for $\mathbf{m}$, mean branch) and $\frac{\partial \tilde{\mathbf{\Sigma}}}{\partial \mathbf{\Sigma}}$ (for $\mathbf{q}$ and $\mathbf{S}$).

A.2. Gradient w.r.t. $\mathbf{m}$: Covariance Branch

Considering the symmetry of Eq. (9), we can define an intermediate variable $\mathbf{T} = \mathbf{W}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}$. Then we can denote:

$$\tilde{\mathbf{\Sigma}} = \mathbf{T}^{\mathrm{T}}\mathbf{\Sigma}\mathbf{T} = \begin{bmatrix} a & b & \text{Skipped} \\ b & c & \text{Skipped} \\ \text{Skipped} & \text{Skipped} & \text{Skipped} \end{bmatrix} \tag{23}$$

where the third row and column are skipped and

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{00} & \mathbf{T}_{01} & \mathbf{T}_{02} \\ \mathbf{T}_{10} & \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{20} & \mathbf{T}_{21} & \mathbf{T}_{22} \end{bmatrix}, \mathbf{\Sigma} = \begin{bmatrix} c_0 & c_1 & c_2 \\ c_1 & c_3 & c_4 \\ c_2 & c_4 & c_5 \end{bmatrix} \tag{24}$$

We can conclude from Appendix A.1 that $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{\Sigma}}}$ has been given by [6], i.e. $\frac{\partial \mathcal{L}}{\partial a}$, $\frac{\partial \mathcal{L}}{\partial b}$, $\frac{\partial \mathcal{L}}{\partial c}$ are known. From Eqs. (23) and (24) we can get:

$$\begin{aligned} a = &(\mathbf{T}_{00}c_0 + \mathbf{T}_{10}c_1 + \mathbf{T}_{20}c_2)\mathbf{T}_{00} \\ &+ (\mathbf{T}_{00}c_1 + \mathbf{T}_{10}c_3 + \mathbf{T}_{20}c_4)\mathbf{T}_{10} \\ &+ (\mathbf{T}_{00}c_2 + \mathbf{T}_{10}c_4 + \mathbf{T}_{20}c_5)\mathbf{T}_{20} \end{aligned} \tag{25}$$

$$\begin{aligned} b = &(\mathbf{T}_{00}c_0 + \mathbf{T}_{10}c_1 + \mathbf{T}_{20}c_2)\mathbf{T}_{01} \\ &+ (\mathbf{T}_{00}c_1 + \mathbf{T}_{10}c_3 + \mathbf{T}_{20}c_4)\mathbf{T}_{11} \\ &+ (\mathbf{T}_{00}c_2 + \mathbf{T}_{10}c_4 + \mathbf{T}_{20}c_5)\mathbf{T}_{21} \end{aligned} \tag{26}$$

$$\begin{aligned} c = &(\mathbf{T}_{01}c_0 + \mathbf{T}_{11}c_1 + \mathbf{T}_{21}c_2)\mathbf{T}_{01} \\ &+ (\mathbf{T}_{01}c_1 + \mathbf{T}_{11}c_3 + \mathbf{T}_{21}c_4)\mathbf{T}_{11} \\ &+ (\mathbf{T}_{01}c_2 + \mathbf{T}_{11}c_4 + \mathbf{T}_{21}c_5)\mathbf{T}_{21} \end{aligned} \tag{27}$$

Then propagate to $\mathbf{T}$:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{00}} &= \frac{\partial\mathcal{L}}{\partial a}\frac{\partial a}{\partial\mathbf{T}_{00}} + \frac{\partial\mathcal{L}}{\partial b}\frac{\partial b}{\partial\mathbf{T}_{00}} + \frac{\partial\mathcal{L}}{\partial c}\frac{\partial c}{\partial\mathbf{T}_{00}} \\ &= 2(\mathbf{T}_{00}c_0 + \mathbf{T}_{10}c_1 + \mathbf{T}_{20}c_2)\frac{\partial\mathcal{L}}{\partial a} \\ &+ (\mathbf{T}_{01}c_0 + \mathbf{T}_{11}c_1 + \mathbf{T}_{21}c_2)\frac{\partial\mathcal{L}}{\partial b} \end{aligned} \tag{28}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{10}} &= 2(\mathbf{T}_{00}c_1 + \mathbf{T}_{10}c_3 + \mathbf{T}_{20}c_4)\frac{\partial\mathcal{L}}{\partial a} \\ &+ (\mathbf{T}_{01}c_1 + \mathbf{T}_{11}c_3 + \mathbf{T}_{21}c_4)\frac{\partial\mathcal{L}}{\partial b}\end{aligned} \tag{29}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{20}} &= 2(\mathbf{T}_{00}c_2 + \mathbf{T}_{10}c_4 + \mathbf{T}_{20}c_5)\frac{\partial\mathcal{L}}{\partial a} \\ &+ (\mathbf{T}_{01}c_2 + \mathbf{T}_{11}c_4 + \mathbf{T}_{21}c_5)\frac{\partial\mathcal{L}}{\partial b}\end{aligned} \tag{30}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{01}} &= 2(\mathbf{T}_{01}c_0 + \mathbf{T}_{11}c_1 + \mathbf{T}_{21}c_2)\frac{\partial\mathcal{L}}{\partial c} \\ &+ (\mathbf{T}_{00}c_0 + \mathbf{T}_{10}c_1 + \mathbf{T}_{20}c_2)\frac{\partial\mathcal{L}}{\partial b}\end{aligned} \tag{31}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{11}} &= 2(\mathbf{T}_{01}c_1 + \mathbf{T}_{11}c_3 + \mathbf{T}_{21}c_4)\frac{\partial\mathcal{L}}{\partial c} \\ &+ (\mathbf{T}_{00}c_1 + \mathbf{T}_{10}c_3 + \mathbf{T}_{20}c_4)\frac{\partial\mathcal{L}}{\partial b}\end{aligned} \tag{32}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{21}} &= 2(\mathbf{T}_{01}c_2 + \mathbf{T}_{11}c_4 + \mathbf{T}_{21}c_5)\frac{\partial\mathcal{L}}{\partial c} \\ &+ (\mathbf{T}_{00}c_2 + \mathbf{T}_{10}c_4 + \mathbf{T}_{20}c_5)\frac{\partial\mathcal{L}}{\partial b}\end{aligned} \tag{33}$$

With the help of $\mathbf{T}$, we can propagate the loss to our $\mathbf{J}$:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{J}_{00}} &= \frac{\partial\mathcal{L}}{\partial\mathbf{T}_{00}}\frac{\partial\mathbf{T}_{00}}{\partial\mathbf{J}_{00}} + \frac{\partial\mathcal{L}}{\partial\mathbf{T}_{10}}\frac{\partial\mathbf{T}_{10}}{\partial\mathbf{J}_{00}} + \frac{\partial\mathcal{L}}{\partial\mathbf{T}_{20}}\frac{\partial\mathbf{T}_{20}}{\partial\mathbf{J}_{00}} \\ &= \mathbf{W}_{00}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{00}} + \mathbf{W}_{01}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{10}} + \mathbf{W}_{02}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{20}}\end{aligned} \tag{34}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{J}_{01}} = 0 \tag{35}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{J}_{02}} = \mathbf{W}_{20}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{00}} + \mathbf{W}_{21}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{10}} + \mathbf{W}_{22}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{20}} \tag{36}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{J}_{10}} = \mathbf{W}_{00}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{01}} + \mathbf{W}_{01}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{11}} + \mathbf{W}_{02}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{21}} \tag{37}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{J}_{11}} = \mathbf{W}_{10}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{01}} + \mathbf{W}_{11}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{11}} + \mathbf{W}_{12}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{21}} \tag{38}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{J}_{12}} = \mathbf{W}_{20}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{01}} + \mathbf{W}_{21}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{11}} + \mathbf{W}_{22}\frac{\partial\mathcal{L}}{\partial\mathbf{T}_{21}} \tag{39}$$

According to Eq. (13), $\mathbf{t}$ is the next to be propagated:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial t_x} &= \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{00}}\frac{\partial\mathbf{J}_{00}}{\partial t_x} + \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{02}}\frac{\partial\mathbf{J}_{02}}{\partial t_x} + \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{10}}\frac{\partial\mathbf{J}_{10}}{\partial t_x} \\ &+ \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{11}}\frac{\partial\mathbf{J}_{11}}{\partial t_x} + \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{12}}\frac{\partial\mathbf{J}_{12}}{\partial t_x}\end{aligned} \tag{40}$$

$$\frac{\partial\mathcal{L}}{\partial t_x} = \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{10}}\frac{\partial\mathbf{J}_{10}}{\partial t_y} + \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{11}}\frac{\partial\mathbf{J}_{11}}{\partial t_y} + \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{12}}\frac{\partial\mathbf{J}_{12}}{\partial t_y} \tag{41}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial t_z} &= \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{00}}\frac{\partial\mathbf{J}_{00}}{\partial t_z} + \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{02}}\frac{\partial\mathbf{J}_{02}}{\partial t_z} + \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{10}}\frac{\partial\mathbf{J}_{10}}{\partial t_z} \\ &+ \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{11}}\frac{\partial\mathbf{J}_{11}}{\partial t_z} + \frac{\partial\mathcal{L}}{\partial\mathbf{J}_{12}}\frac{\partial\mathbf{J}_{12}}{\partial t_z}\end{aligned} \tag{42}$$

From Eqs. (17) to (22), we can get the second derivative of the equirectangular camera model:

$$\frac{\partial\mathbf{J}_{00}}{\partial t_x} = -\frac{W}{2\pi}\frac{2t_xt_z}{(t_x^2+t_z^2)^2} \tag{43}$$

$$\frac{\partial\mathbf{J}_{02}}{\partial t_x} = +\frac{W}{2\pi}\frac{t_x^2-t_z^2}{(t_x^2+t_z^2)^2} \tag{44}$$

$$\frac{\partial\mathbf{J}_{10}}{\partial t_x} = +\frac{H}{\pi}\frac{t_y[2t_x^2(t_x^2+t_z^2)-t_z^2t_r^2]}{t_r^4(t_x^2+t_z^2)\sqrt{t_x^2+t_z^2}} \tag{45}$$

$$\frac{\partial\mathbf{J}_{11}}{\partial t_x} = +\frac{H}{\pi}\frac{t_x[t_y^2-(t_x^2+t_z^2)]}{t_r^4\sqrt{t_x^2+t_z^2}} \tag{46}$$

$$\frac{\partial\mathbf{J}_{12}}{\partial t_x} = +\frac{H}{\pi}\frac{t_xt_yt_z[t_r^2+2(t_x^2+t_z^2)]}{t_r^4(t_x^2+t_z^2)\sqrt{t_x^2+t_z^2}} \tag{47}$$

$$\frac{\partial\mathbf{J}_{10}}{\partial t_y} = +\frac{H}{\pi}\frac{t_x[t_y^2-(t_x^2+t_z^2)]}{t_r^4\sqrt{t_x^2+t_z^2}} \tag{48}$$

$$\frac{\partial\mathbf{J}_{11}}{\partial t_y} = -\frac{H}{\pi}\frac{2t_y\sqrt{t_x^2+t_z^2}}{t_r^4} \tag{49}$$

$$\frac{\partial\mathbf{J}_{12}}{\partial t_y} = +\frac{H}{\pi}\frac{t_z[t_y^2-(t_x^2+t_z^2)]}{t_r^4\sqrt{t_x^2+t_z^2}} \tag{50}$$

$$\frac{\partial\mathbf{J}_{00}}{\partial t_z} = +\frac{W}{2\pi}\frac{t_x^2-t_z^2}{(t_x^2+t_z^2)^2} \tag{51}$$

$$\frac{\partial\mathbf{J}_{02}}{\partial t_z} = +\frac{W}{2\pi}\frac{2t_xt_z}{(t_x^2+t_z^2)^2} \tag{52}$$

$$\frac{\partial\mathbf{J}_{10}}{\partial t_z} = +\frac{H}{\pi}\frac{t_xt_yt_z[t_r^2+2(t_x^2+t_z^2)]}{t_r^4(t_x^2+t_z^2)\sqrt{t_x^2+t_z^2}} \tag{53}$$

$$\frac{\partial\mathbf{J}_{11}}{\partial t_z} = +\frac{H}{\pi}\frac{t_z[t_y^2-(t_x^2+t_z^2)]}{t_r^4\sqrt{t_x^2+t_z^2}} \tag{54}$$

$$\frac{\partial\mathbf{J}_{12}}{\partial t_z} = +\frac{H}{\pi}\frac{t_y[2t_z^2(t_x^2+t_z^2)-t_x^2t_r^2]}{t_r^4(t_x^2+t_z^2)\sqrt{t_x^2+t_z^2}} \tag{55}$$

Note that there are several duplicated parts in the second derivative, and we could precompute them to accelerate the optimization. By substituting the above derivative into Eqs. (40) to (42), we propagate the gradient to $\mathbf{t}$.

Finally, propagate to $\mathbf{m}$ according to Eq. (10). Suppose we have completed the color branch in Eq. (4) and got a vector storing its gradients w.r.t. $\mathbf{m}$. We could simply accumulate the gradient from another branch:

$$\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial m_x} \\ \frac{\partial \mathcal{L}}{\partial m_y} \\ \frac{\partial \mathcal{L}}{\partial m_z} \end{bmatrix} += \mathbf{W}^{\mathrm{T}} \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial t_x} \\ \frac{\partial \mathcal{L}}{\partial t_y} \\ \frac{\partial \mathcal{L}}{\partial t_z} \end{bmatrix} \tag{56}$$

A.3. Gradient w.r.t. $\mathbf{m}$: Gaussian Mean Branch

From Appendix A.1, we know $\frac{\partial \mathcal{L}}{\partial \mathbf{p}} = \left[\frac{\partial \mathcal{L}}{\partial p_x}, \frac{\partial \mathcal{L}}{\partial p_y}\right]^{\mathrm{T}}$ has been given. We compute and record the results of Eqs. (17) to (22), and further propagate the gradient to $\mathbf{t}$ (a new branch of gradient w.r.t. $\mathbf{t}$, different from Eqs. (40) to (42)):

$$\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial t_x} \\ \frac{\partial \mathcal{L}}{\partial t_y} \\ \frac{\partial \mathcal{L}}{\partial t_z} \end{bmatrix} \text{(mean branch)} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial p_x}\frac{\partial p_x}{\partial t_x} + \frac{\partial \mathcal{L}}{\partial p_y}\frac{\partial p_y}{\partial t_x} \\ \frac{\partial \mathcal{L}}{\partial p_x}\frac{\partial p_x}{\partial t_y} + \frac{\partial \mathcal{L}}{\partial p_y}\frac{\partial p_y}{\partial t_y} \\ \frac{\partial \mathcal{L}}{\partial p_x}\frac{\partial p_x}{\partial t_z} + \frac{\partial \mathcal{L}}{\partial p_y}\frac{\partial p_y}{\partial t_z} \end{bmatrix} \tag{57}$$

Again, we accumulate this branch to the final $\frac{\partial \mathcal{L}}{\partial \mathbf{m}}$:

$$\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial m_x} \\ \frac{\partial \mathcal{L}}{\partial m_y} \\ \frac{\partial \mathcal{L}}{\partial m_z} \end{bmatrix} += \mathbf{W}^{\mathrm{T}} \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial t_x} \\ \frac{\partial \mathcal{L}}{\partial t_y} \\ \frac{\partial \mathcal{L}}{\partial t_z} \end{bmatrix} \text{(mean branch)} \tag{58}$$

Parameter	Value
Position l.r. (Initial)	0.00016
Position l.r. (Final)	0.0000016
Position l.r. delay multiplier	0.01
Position l.r. max. dumping steps	30000
Feature l.r.	0.0025
Opacity l.r.	0.05
Scaling l.r.	0.005
Rotation l.r.	0.001

Table 1. Setup of Learning Rates

A.4. Gradient w.r.t. $\mathbf{q}$ and $\mathbf{S}$

Recall Eqs. (23) to (27). This time we fix $\mathbf{T} = \mathbf{W}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}$ in order to propagate the gradient to $\mathbf{\Sigma}$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial c_0} &= \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial c_0} + \frac{\partial \mathcal{L}}{\partial b}\frac{\partial b}{\partial c_0} + \frac{\partial \mathcal{L}}{\partial c}\frac{\partial c}{\partial c_0} \\ &= \mathbf{T}_{00}^2\frac{\partial \mathcal{L}}{\partial a} + \mathbf{T}_{00}\mathbf{T}_{01}\frac{\partial \mathcal{L}}{\partial b} + \mathbf{T}_{01}^2\frac{\partial \mathcal{L}}{\partial c} \end{aligned} \tag{59}$$

$$\frac{\partial \mathcal{L}}{\partial c_3} = \mathbf{T}_{10}^2\frac{\partial \mathcal{L}}{\partial a} + \mathbf{T}_{10}\mathbf{T}_{11}\frac{\partial \mathcal{L}}{\partial b} + \mathbf{T}_{11}^2\frac{\partial \mathcal{L}}{\partial c} \tag{60}$$

$$\frac{\partial \mathcal{L}}{\partial c_5} = \mathbf{T}_{20}^2\frac{\partial \mathcal{L}}{\partial a} + \mathbf{T}_{20}\mathbf{T}_{21}\frac{\partial \mathcal{L}}{\partial b} + \mathbf{T}_{21}^2\frac{\partial \mathcal{L}}{\partial c} \tag{61}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial c_1} &= 2\mathbf{T}_{00}\mathbf{T}_{10}\frac{\partial \mathcal{L}}{\partial a} + 2\mathbf{T}_{01}\mathbf{T}_{11}\frac{\partial \mathcal{L}}{\partial c} \\ &+ (\mathbf{T}_{10}\mathbf{T}_{01} + \mathbf{T}_{00}\mathbf{T}_{11})\frac{\partial \mathcal{L}}{\partial b} \end{aligned} \tag{62}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial c_2} &= 2\mathbf{T}_{00}\mathbf{T}_{20}\frac{\partial \mathcal{L}}{\partial a} + 2\mathbf{T}_{01}\mathbf{T}_{21}\frac{\partial \mathcal{L}}{\partial c} \\ &+ (\mathbf{T}_{20}\mathbf{T}_{01} + \mathbf{T}_{00}\mathbf{T}_{21})\frac{\partial \mathcal{L}}{\partial b} \end{aligned} \tag{63}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial c_4} &= 2\mathbf{T}_{10}\mathbf{T}_{20}\frac{\partial \mathcal{L}}{\partial a} + 2\mathbf{T}_{11}\mathbf{T}_{21}\frac{\partial \mathcal{L}}{\partial c} \\ &+ (\mathbf{T}_{20}\mathbf{T}_{11} + \mathbf{T}_{10}\mathbf{T}_{21})\frac{\partial \mathcal{L}}{\partial b} \end{aligned} \tag{64}$$

The next steps, including $\frac{\partial \mathbf{\Sigma}}{\partial \mathbf{R}}\frac{\partial \mathbf{R}}{\partial \mathbf{q}}$ and $\frac{\partial \mathbf{\Sigma}}{\partial \mathbf{S}}$, are in the 3D world coordinate system so not related to the camera model. We could continue to use the derivation given by [6].

B. Hyperparameter Setup

B.1. Learning Rate

We keep all learning rates (l.r.) the same as the original 3DGS [6], including the continuous dumping process (determined by the delay multiplier and maximum number of dumping steps) of the position learning rate. Specific values are shown in Tab. 1.

Parameter	Value	Explanation
λ_{DSSIM}	0.2	Weight of DSSIM part in the photorealistic loss.
Densification Interval	100	Densify the Gaussians every certain iterations.
Opacity Reset Interval	3000	Reset all opacity every certain iterations.
Densify Minimum Opacity	0.005	Prune Gaussians whose opacity is less than the threshold after each densification.
Densify from Iteration	500	Only densify after a certain iteration.
Densify until Iteration	15000	Only densify and reset opacity before a certain iteration.
Densify Gradient Threshold	0.0002	Only densify Gaussians whose 2D position gradient is not less than the threshold.
Prune by Extent	False (EgoNeRF) True (Others)	If true, prune Gaussians whose scale is too large compared with the scene extent.
Percent Dense	0.01	Deciding a threshold. If a Gaussian is to be densified, it is split if its scale is greater than the threshold, otherwise cloned.

Table 2. Setup of Densification Hyperparameters

B.2. Densification

As shown in Tab. 2, we also keep all hyperparameters related to densification the same with [6], except for pruning by scene extent or not. We disable this feature only on EgoNeRF dataset [3], because it applies an egocentric camera motion pattern and the magnitude of motion between frames is too small (only a few centimeters) compared to the actual scene extent (a few meters or even larger). Scale of Gaussians easily exceeds the threshold estimated from camera poses. They are pruned very soon after beginning and leave a completely empty scene behind. We observe no extra floaters or blurs after disabling it on the EgoNeRF dataset, so we apply this setup.

C. Performance Compared with Concurrent 3DGS-Based Work

OP43DGS [5] is a work concurrent with OmniGS. It leverages function optimization theory to analyze the function's minima, providing an optimal projection strategy for 3DGS, which can accommodate a variety of camera models, and thus supports training from omnidirectional inputs.

We ran OP43DGS on both the 360Roam [4] and EgoNeRF [3] dataset with the same RTX 3090 GPU. We used exactly the same inputs as we used for our method.

The overall results are reported in Tab. 3, indicating that while we achieved similar reconstruction quality in most

Dataset	Method	OP43DGS [5]	Ours
360Roam	PSNR↑	25.441	**25.464**
	SSIM↑	**0.810**	0.806
	LPIPS↓	0.159	**0.141**
	FPS↑	14	**121**
OmniBlender-Indoor	PSNR↑	34.718	**35.330**
	SSIM↑	**0.923**	0.917
	LPIPS↓	0.075	**0.072**
	FPS↑	9	**115**
OmniBlender-Outdoor	PSNR↑	32.319	**32.670**
	SSIM↑	**0.928**	0.919
	LPIPS↓	0.049	**0.044**
	FPS↑	2	**116**
Ricoh360	PSNR↑	24.135	**26.032**
	SSIM↑	0.766	**0.825**
	LPIPS↓	0.234	**0.128**
	FPS↑	<1	**91**

Table 3. Overall quantitative evaluation results compared to OP43DGS. FPS means the novel-view rendering FPS after training. We mark the best results with **first**.

scenes (PSNR+0.023, SSIM−0.004, LPIPS−0.018 on 360Roam, PSNR+0.612, SSIM−0.006, LPIPS−0.003 on EgoNeRF-OmniBlender-Indoor, PSNR+0.351, SSIM−0.009, LPIPS−0.005 on EgoNeRF-OmniBlender-Outdoor, PSNR+1.897, SSIM+0.059, LPIPS−0.106 on EgoNeRF-Ricoh360), our method rendered new views much faster than OP43DGS (FPS+107 on 360Roam, FPS+106 on EgoNeRF-OmniBlender-Indoor, FPS+114 on EgoNeRF-OmniBlender-Outdoor, FPS+90 on EgoNeRF-Ricoh360), indicating that the algorithm proposed by us had less computational complexity than OP43DGS.

Training time and per-scene results are reported in the tables of Appendix E. Note that in some real-world outdoor Ricoh360 scenes, OP43DGS threw *CUDA out of memory* errors at the early stage of optimization, so we report the results evaluated just a few earlier than the errors occurred in Tab. 9. This also caused its performance degradation. An example of this phenomenon is shown in Supple. Fig. 1.

D. Runtime Performance

The runtime peak allocated GPU memory and the reconstructed model file size are reported in Tab. 5 (EgoNeRF dataset [3] OmniBlender scenes), Tab. 6 (EgoNeRF dataset Ricoh360 scenes) and Tab. 7 (360Roam dataset [4]).

We also conducted an experiment on performance w.r.t. hyperparameter *densify gradient threshold* which controlled the level of densification. Results are reported in Tab. 4. All experiments were run to 32k iterations. When we suppress densification by increasing the threshold, the system consumed less GPU memory and storage, reaching faster training (saving about 10 minutes per scene, compared with the main paper experiments) and higher novel-view rendering FPS, at the cost of quality loss.

E. Per-Scene Results

We additionally list the per-scene evaluation results in Tab. 8 (EgoNeRF dataset [3] OmniBlender scenes), Tab. 9 (EgoNeRF dataset Ricoh360 scenes) and Tab. 10 (360Roam dataset [4]),. We gather the reported results for baselines in the way stated in the body of our paper. For OP43DGS [5], we ran the experiments on our RTX 3090 machine as stated in Appendix C, since their paper did not provide data on 360Roam or EgoNeRF dataset.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, 2022. 9, 10, 11

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, pages 333–350, 2022. 9, 10, 11

[3] Changwoon Choi, Sang Min Kim, and Young Min Kim. Balanced spherical grid for egocentric view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16590–16599, 2023. 5, 6, 9, 10

[4] Huang Huajian, Chen Yingshu, Zhang Tianjia, and Yeung Sai-Kit. 360Roam: Real-time indoor roaming using geometry-aware 360° radiance fields. *arXiv preprint, arXiv:2208.02705*, 2022. 5, 6, 11

[5] Letian Huang, Jiayang Bai, Jie Guo, Yuanqi Li, and Yanwen Guo. On the error analysis of 3d gaussian splatting and an optimal projection strategy. In *European Conference on Computer Vision (ECCV)*, 2024. 5, 6, 7, 8, 9, 10, 11

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1, 2, 4, 5, 8

[7] 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 *European Conference on Computer Vision (ECCV)*, 2020. 9, 10, 11

[8] Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. OpenMVG: Open multiple view geometry. In *Reproducible Research in Pattern Recognition*, pages 60–74, 2017. 2

[9] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4), 2022. 11

[10] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 2

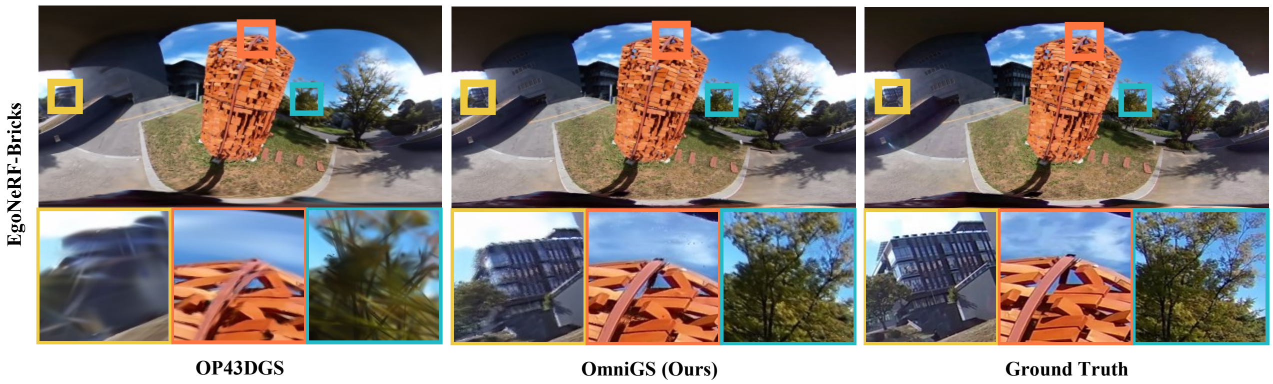

Figure 1. An example of OP43DGS [5] underfitting degradation caused by *CUDA out of memory* on EgoNeRF-Ricoh360 dataset.

Scene	Densify Gradient Threshold	0.0002 (Main Paper)	0.001	0.002
360Roam-*bar*	PSNR↑	22.471	22.300	22.285
	FPS↑	69	196	219
	GPU Mem. (MB)	4093.77	3004.83	3000.83
	Model Size (MB)	252.44	62.41	55.17
OmniBlender-Indoor-*barbershop*	PSNR↑	36.847	34.050	33.014
	FPS↑	114	195	208
	GPU Mem. (MB)	1941.37	1436.10	1407.39
	Model Size (MB)	92.31	14.43	10.11
OmniBlender-Outdoor-*bistro_bike*	PSNR↑	37.915	33.136	30.993
	FPS↑	117	170	197
	GPU Mem. (MB)	2006.38	1529.46	1468.28
	Model Size (MB)	91.37	23.00	18.37
Ricoh360-*bricks*	PSNR↑	24.664	23.187	22.214
	FPS↑	72	212	310
	GPU Mem. (MB)	3251.13	1934.14	1835.28
	Model Size (MB)	222.18	13.48	4.37

Table 4. Performance w.r.t. gradient threshold of densification in representative scenes. Larger threshold leads to less densification. GPU Mem. means peak allocated GPU memory. Model is the output *.ply file. Note that we use 1MB = 1024KB = 1048576Bytes.

Scene (Indoor)	archiviz-flat	barbershop	classroom	restroom			
GPU Mem. (MB)	2036.20	1941.37	1993.29	2090.24			
Model Size (MB)	104.75	92.31	98.85	103.16			
Scene (Outdoor)	bistro_bike	bistro_square	fisher-hut	lone_monk	LOU	pavilion_midday_chair	pavilion_midday_pond
GPU Mem. (MB)	2006.38	2191.48	1929.33	2038.89	1733.18	1879.11	2350.89
Model Size (MB)	91.37	123.26	65.83	100.39	46.37	70.87	121.28

Table 5. Per-scene GPU memory and storage usage results on EgoNeRF-OmniBlender dataset. GPU Mem. means peak allocated GPU memory. Model is the output *.ply file. Note that we use 1MB = 1024KB = 1048576Bytes.

Scene	bricks	bridge	bridge_under	cat_tower	center	farm
GPU Mem. (MB)	3251.13	3072.84	3404.03	2716.56	2562.45	3319.10
Model Size (MB)	222.18	178.90	199.44	129.40	82.06	227.30
Scene	flower	gallery_chair	gallery_pillar	garden	poster	
GPU Mem. (MB)	2941.10	2533.39	2481.25	2667.72	3599.49	
Model Size (MB)	157.52	76.79	85.55	118.96	140.65	

Table 6. Per-scene GPU memory and storage usage results on EgoNeRF-Ricoh360 dataset. GPU Mem. means peak allocated GPU memory. Model is the output *.ply file. Note that we use 1MB = 1024KB = 1048576Bytes.

Scene	Usage	3DGS [6]	OP43DGS [5]	Ours
Bar	GPU Mem. (MB)	15073.06	4931.10	4093.77
	Model Size (MB)	381.88	217.07	252.44
Base	GPU Mem. (MB)	15005.90	5035.24	4204.85
	Model Size (MB)	399.02	257.82	278.87
Cafe	GPU Mem. (MB)	8834.68	4120.48	2796.73
	Model Size (MB)	361.73	209.94	220.08
Canteen	GPU Mem. (MB)	6889.46	2604.35	2245.97
	Model Size (MB)	208.74	140.69	154.60
Center	GPU Mem. (MB)	12168.91	3999.04	3366.32
	Model Size (MB)	140.79	143.10	177.16
Corridor	GPU Mem. (MB)	5505.35	2108.64	1805.57
	Model Size (MB)	133.39	88.59	95.37
Innovation	GPU Mem. (MB)	15670.58	5427.03	4573.49
	Model Size (MB)	323.34	246.66	301.42
Lab	GPU Mem. (MB)	8956.87	3750.96	2575.33
	Model Size (MB)	267.08	130.42	153.75
Library	GPU Mem. (MB)	7184.92	2831.33	2202.61
	Model Size (MB)	215.66	116.89	131.17
Office	GPU Mem. (MB)	9748.92	3407.26	2563.91
	Model Size (MB)	146.47	106.09	113.45

Table 7. Per-scene GPU memory and storage usage results (32k iterations, around 25 minutes) on the 360Roam dataset. GPU Mem. means peak allocated GPU memory. Model is the output *.ply file. Note that we use $1\text{MB} = 1024\text{KB} = 1048576\text{Bytes}$. We additionally report the results of 3DGS [6] (perspective 1024×1024 inputs, 28k iterations, around 25 minutes) and OP43DGS [5] (panoramic 712×1520 inputs, the same as ours, but run for 12k iterations, around 40 minutes) for comparison.

Scene	Method Iterations Training Time	NeRF [7] 10k $>$ 5 hours	Mip-NeRF 360 [1] 10k $>$ 2 hours	TensoRF [2] 100k $\approx$ 40 min	EgoNeRF [3] 10k $\approx$ 30 min	OP43DGS [5] 10k $>$ 30 min[1]	Ours 32k $\approx$ 25 min
archiviz-flat	PSNR↑	27.460	28.760	31.000	30.480	35.254	35.496
	SSIM↑	0.820	0.848	0.871	0.876	0.959	0.955
	LPIPS↓	0.333	0.275	0.209	0.189	0.031	0.026
barbershop	PSNR↑	28.010	28.350	30.200	32.530	35.431	36.847
	SSIM↑	0.838	0.845	0.887	0.930	0.968	0.965
	LPIPS↓	0.324	0.346	0.237	0.128	0.029	0.025
classroom	PSNR↑	26.750	24.500	28.910	27.470	32.773	32.942
	SSIM↑	0.732	0.724	0.782	0.794	0.870	0.862
	LPIPS↓	0.491	0.482	0.410	0.323	0.142	0.142
restroom	PSNR↑	28.410	28.030	26.910	30.430	35.414	36.035
	SSIM↑	0.636	0.637	0.624	0.761	0.896	0.888
	LPIPS↓	0.551	0.544	0.647	0.350	0.098	0.096
bistro_bike	PSNR↑	21.500	25.270	23.550	31.290	36.775	37.915
	SSIM↑	0.594	0.764	0.668	0.930	0.975	0.971
	LPIPS↓	0.562	0.299	0.468	0.074	0.019	0.014
bistro_square	PSNR↑	18.640	21.820	20.500	24.520	28.437	28.904
	SSIM↑	0.532	0.723	0.608	0.862	0.949	0.948
	LPIPS↓	0.678	0.303	0.444	0.126	0.038	0.028
fisher-hut	PSNR↑	27.900	29.020	29.590	30.010	32.828	31.949
	SSIM↑	0.747	0.768	0.770	0.788	0.875	0.855
	LPIPS↓	0.490	0.418	0.424	0.281	0.081	0.081
lone_monk	PSNR↑	23.900	25.180	24.640	29.280	34.224	34.960
	SSIM↑	0.714	0.777	0.735	0.901	0.964	0.963
	LPIPS↓	0.361	0.266	0.299	0.115	0.034	0.023
LOU	PSNR↑	25.490	27.810	31.350	32.010	36.695	37.309
	SSIM↑	0.786	0.852	0.906	0.914	0.965	0.957
	LPIPS↓	0.345	0.257	0.155	0.095	0.031	0.025
pavilion_midday_chair	PSNR↑	26.050	26.850	27.700	29.860	32.144	31.965
	SSIM↑	0.800	0.809	0.810	0.905	0.952	0.931
	LPIPS↓	0.302	0.297	0.269	0.099	0.035	0.038
pavilion_midday_pond	PSNR↑	21.940	23.030	22.430	24.680	25.133	25.685
	SSIM↑	0.627	0.686	0.641	0.774	0.817	0.809
	LPIPS↓	0.468	0.301	0.347	0.164	0.102	0.101

Table 8. Per-scene quantitative evaluation results on EgoNeRF-OmniBlender dataset, min denotes minutes. The first 4 are indoor scenes, the other 7 are outdoor scenes. [1]The time OP43DGS took for 10k iterations varied from 30 min to more than 4 hours in different scenes.

Scene	Method Iterations Training Time	NeRF [7] 10k $>$ 5 hours	Mip-NeRF 360 [1] 10k $>$ 2 hours	TensoRF [2] 100k $\approx$ 40 min	EgoNeRF [3] 10k $\approx$ 30 min	OP43DGS [5] 6k[1] $>$ 30 min	Ours 32k $\approx$ 25 min
bricks	PSNR↑	20.640	22.080	23.080	22.680	21.112	24.664
	SSIM↑	0.594	0.676	0.701	0.720	0.713	0.836
	LPIPS↓	0.547	0.371	0.342	0.292	0.277	0.120
bridge	PSNR↑	21.480	22.730	23.270	22.980	23.297	23.676
	SSIM↑	0.634	0.695	0.695	0.713	0.781	0.790
	LPIPS↓	0.505	0.363	0.360	0.312	0.150	0.136
bridge_under	PSNR↑	22.430	23.370	24.560	24.250	23.678	26.601
	SSIM↑	0.650	0.723	0.736	0.763	0.779	0.873
	LPIPS↓	0.499	0.390	0.332	0.282	0.231	0.089
cat_tower	PSNR↑	22.180	23.380	23.840	23.690	22.930	24.756
	SSIM↑	0.615	0.668	0.665	0.681	0.695	0.773
	LPIPS↓	0.610	0.460	0.487	0.380	0.311	0.159
center	PSNR↑	25.810	27.730	29.250	28.070	28.501	29.165
	SSIM↑	0.783	0.838	0.849	0.850	0.885	0.887
	LPIPS↓	0.484	0.293	0.279	0.236	0.111	0.100
farm	PSNR↑	20.290	21.660	22.020	21.980	20.717	22.207
	SSIM↑	0.549	0.626	0.631	0.651	0.650	0.726
	LPIPS↓	0.554	0.366	0.378	0.322	0.342	0.166
flower	PSNR↑	19.520	20.930	21.720	21.510	20.392	22.300
	SSIM↑	0.523	0.593	0.595	0.617	0.620	0.724
	LPIPS↓	0.698	0.517	0.530	0.424	0.402	0.191
gallery_chair	PSNR↑	25.600	27.030	28.040	27.130	27.477	28.790
	SSIM↑	0.783	0.823	0.831	0.834	0.870	0.892
	LPIPS↓	0.538	0.401	0.385	0.323	0.166	0.105
gallery_pillar	PSNR↑	25.300	26.970	28.140	27.500	27.067	28.731
	SSIM↑	0.769	0.821	0.831	0.835	0.856	0.884
	LPIPS↓	0.414	0.275	0.274	0.227	0.132	0.087
garden	PSNR↑	24.490	26.090	26.470	26.500	25.548	27.133
	SSIM↑	0.653	0.695	0.692	0.713	0.738	0.795
	LPIPS↓	0.562	0.427	0.457	0.361	0.247	0.155
poster	PSNR↑	22.790	25.110	26.380	25.570	24.763	28.333
	SSIM↑	0.742	0.816	0.832	0.831	0.841	0.898
	LPIPS↓	0.509	0.357	0.314	0.290	0.204	0.104

Table 9. Per-scene quantitative evaluation results on EgoNeRF-Ricoh360 dataset, min denotes minutes. [1]In some scenes OP43DGS triggered a *CUDA out of memory* error before reaching 6k iterations. This sometimes caused the actual training time to be less than 30 minutes. For the scenes ran out of GPU memory at the early stage of optimization, i.e. *bricks, bridge_under, cat_tower, farm, flower, gallery_chair, gallery_pillar, garden, poster*, we report the OP43DGS results evaluated just a few time before the error occurred.

Scene	Method	NeRF [7]	Mip-NeRF 360 [1]	TensoRF [2]	Instant-NGP [9]	360Roam [4]	OP43DGS[1] [5]	Ours
Bar	PSNR↑	19.049	21.112	21.517	15.163	21.676	22.234	22.471
	SSIM↑	0.601	0.683	0.707	0.488	0.711	0.769	0.771
	LPIPS↓	0.409	0.324	0.294	0.571	0.235	0.192	0.172
Base	PSNR↑	21.255	23.178	13.256	15.668	24.093	24.794	24.853
	SSIM↑	0.598	0.692	0.412	0.472	0.725	0.800	0.796
	LPIPS↓	0.400	0.309	0.791	0.616	0.210	0.134	0.106
Cafe	PSNR↑	20.732	23.508	13.699	17.051	21.969	25.047	25.053
	SSIM↑	0.656	0.746	0.506	0.554	0.720	0.828	0.821
	LPIPS↓	0.378	0.277	0.608	0.521	0.230	0.124	0.109
Canteen	PSNR↑	19.941	21.851	17.886	15.851	21.984	22.500	22.207
	SSIM↑	0.613	0.690	0.605	0.506	0.680	0.750	0.735
	LPIPS↓	0.427	0.356	0.498	0.549	0.303	0.212	0.209
Center	PSNR↑	22.439	24.841	14.391	16.566	25.109	25.381	25.152
	SSIM↑	0.699	0.771	0.530	0.590	0.775	0.818	0.808
	LPIPS↓	0.358	0.280	0.783	0.597	0.226	0.196	0.167
Corridor	PSNR↑	25.342	28.442	14.523	17.722	28.812	28.176	27.798
	SSIM↑	0.754	0.834	0.588	0.618	0.832	0.861	0.848
	LPIPS↓	0.238	0.166	0.676	0.425	0.145	0.106	0.111
Innovation	PSNR↑	22.482	25.433	12.716	17.121	26.191	26.046	25.885
	SSIM↑	0.667	0.754	0.428	0.519	0.771	0.815	0.806
	LPIPS↓	0.324	0.253	0.799	0.543	0.187	0.162	0.115
Lab	PSNR↑	24.135	25.830	15.542	19.075	27.667	27.983	28.023
	SSIM↑	0.763	0.833	0.593	0.651	0.855	0.889	0.886
	LPIPS↓	0.239	0.192	0.642	0.414	0.116	0.083	0.075
Library	PSNR↑	23.909	25.971	13.409	18.450	26.127	26.616	26.345
	SSIM↑	0.656	0.714	0.468	0.519	0.722	0.774	0.767
	LPIPS↓	0.326	0.288	0.836	0.558	0.225	0.193	0.194
Office	PSNR↑	25.149	25.619	13.409	17.521	26.977	25.632	26.848
	SSIM↑	0.714	0.763	0.468	0.563	0.811	0.798	0.817
	LPIPS↓	0.290	0.245	0.836	0.524	0.145	0.184	0.148

Table 10. Per-scene Quantitative evaluation results on the 360Roam dataset. [1]OP43DGS is evaluated at 12k iterations, taking around 40 minutes. Evaluation timing of the other methods are reported in the main paper.