Local Gaussian Density Mixtures for Unstructured Lumigraph Rendering

XIUCHAO WU, State Key Lab of CAD&CG, Zhejiang University, China
JIAMIN XU, Hangzhou Dianzi University, China
CHI WANG, State Key Lab of CAD&CG, Zhejiang University, China
YIFAN PENG, The University of Hong Kong, China
QIXING HUANG, University of Texas at Austin, USA
JAMES TOMPKIN, Brown University, USA
WEIWEI XU*, State Key Lab of CAD&CG, Zhejiang University, China

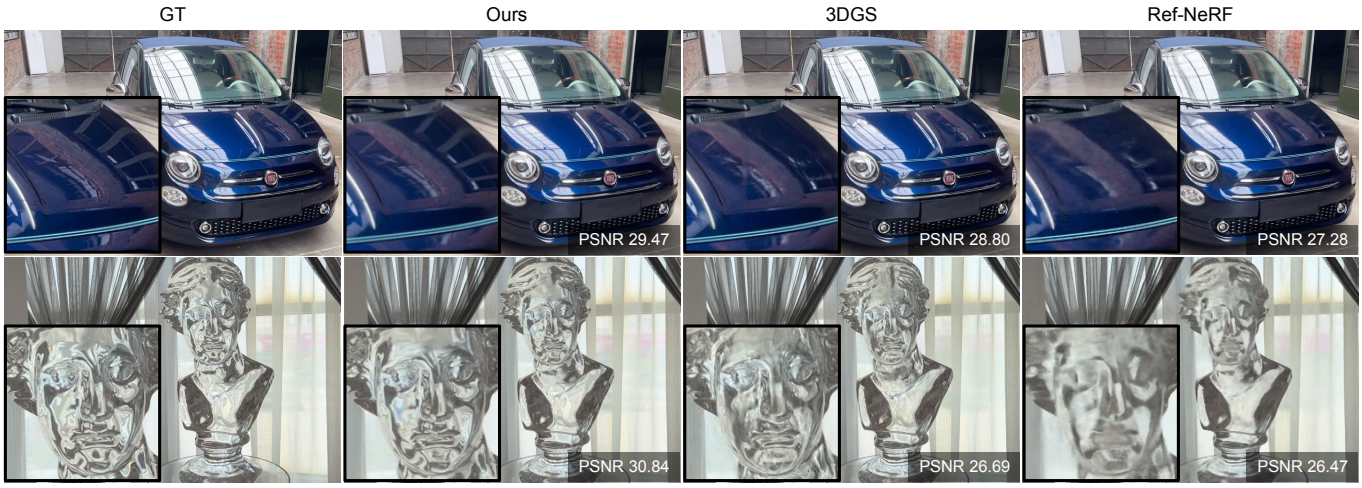

Fig. 1. Image-based rendering is particularly challenging due to complex reflections from curved surfaces and refractions in transparent materials. Compared to existing methods (3DGS [Kerbl et al. 2023] and Ref-NeRF [Verbin et al. 2022]), our approach more accurately reproduces high-frequency view-dependent appearance, such as the reflected building on the car hood in *Blue Car* (top), and the multi-bounce refractions in *Glass Bust* (bottom).

To improve novel view synthesis of curved-surface reflections and refractions, we revisit local geometry-guided ray interpolation techniques with modern differentiable rendering and optimization. In contrast to depth or mesh geometries, our approach uses a local or per-view density represented as Gaussian mixtures along each ray. To synthesize novel views, we warp and fuse local volumes, then alpha-composite using input photograph ray colors from a small set of neighboring images. For fusion, we use a neural blending weight from a shallow MLP. We optimize the local Gaussian density mixtures using both a reconstruction loss and a consistency loss. The consistency loss, based on per-ray KL-divergence, encourages more accurate geometry reconstruction. In scenes with complex reflections captured in our LGDM dataset, the experimental results show that our method outperforms state-of-the-art novel view synthesis methods by 12.2%–37.1% in PSNR, due to its ability to maintain sharper view-dependent appearances. Project webpage: https://xchaowu.github.io/papers/lgdm/index.html

CCS Concepts: • **Computing methodologies** → **Image-based rendering**; *Reflectance modeling*; *Reconstruction.*

Additional Key Words and Phrases: Novel view synthesis, Gaussian mixtures.

ACM Reference Format:
Xiuchao Wu, Jiamin Xu, Chi Wang, Yifan Peng, Qixing Huang, James Tompkin, and Weiwei Xu. 2024. Local Gaussian Density Mixtures for Unstructured Lumigraph Rendering. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–6, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3680528.3687659

*Corresponding author

Authors' addresses: Xiuchao Wu, wuxiuchao@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China; Jiamin Xu, superxjm@yeah.net, Hangzhou Dianzi University, China; Chi Wang, wangchi1995@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China; Yifan Peng, evanpeng@hku.hk, The University of Hong Kong, China; Qixing Huang, huangqx@cs.utexas.edu, University of Texas at Austin, USA; James Tompkin, james_tompkin@brown.edu, Brown University, USA; Weiwei Xu, xww@cad.zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China.

SA Conference Papers '24, December 3–6, 2024, Tokyo, Japan

ACM ISBN 979-8-4007-1131-2/24/12
https://doi.org/10.1145/3680528.3687659

1 INTRODUCTION

There is a spectrum of approaches to novel view synthesis, from local ray interpolation to image-based rendering to global radiance fields. Local ray interpolation methods, popularized with light fields or lumigraphs, interpolate novel ray colors from input photographs via

two-plane ray parameterization [Gortler et al. 1996; Levoy and Hanrahan 1996]. For accurate scene reproduction, local ray interpolation methods require capturing the light field with a high spatio-angular resolution. To reduce this capture burden, unstructured lumigraphs use geometric proxies to guide the selection of corresponding rays to interpolate. Such proxies can be global [Buehler et al. 2001; Eisemann et al. 2008] or per input view [Hedman et al. 2018, 2016]. Rendering quality is determined by the accuracy of the geometric proxy, which itself must be reconstructed. Beyond interpolating input colors, we can also create novel views by reconstructing a global scene representation of color and geometry, such as a neural radiance field (NeRFs) [Mildenhall et al. 2020] or a set of 3D Gaussians (3DGS) [Kerbl et al. 2023]. These representations are optimized by minimizing the difference between the input photographs and their reproduction through volumetric rendering.

One challenging visual phenomenon is reflections that appear to move with camera motion; a second is the related phenomenon of refractions through transparent objects. Many methods, including NeRF and 3DGS, often handle reflections by optimizing virtual geometry to be 'behind' surfaces at the total reflected light path length. For planar reflectors, a global virtual reflection geometry provides a plausible explanation of the visual phenomenon that is consistent with all input photographs [Sinha et al. 2012]. However, for curved reflectors, it is difficult to maintain a consistent virtual geometry, resulting in blurred reflections (Fig. 1). To better model curved reflections, methods have adopted additional physical modeling through surface normal estimation and material decomposition [Verbin et al. 2022; Zhang et al. 2021b]. Such approaches may assume distant environmental lighting conditions [Verbin et al. 2022] that limit application to specific real-world scenarios. Factorizing materials increases the number of free parameters, leading to increased ambiguity during optimization [Zhang et al. 2021b]. As a result, dealing with curved reflective surfaces or transparent objects still poses challenges and producing sharp reflections is difficult.

Given this difficulty, we revisit ideas from geometry-guided ray interpolation techniques with modern neural fields, differentiable rendering, and end-to-end optimization. Interpolation-based methods can avoid the blurring that occurs in global optimization of view-dependent appearance like reflections because only a few similar neighboring photographs are used to produce ray colors. But the appearance still depends significantly on the proxy geometry, and we know that consistent recovery of reflector geometry is tricky for curved reflectors. Rather than a global proxy geometry, we propose for each input view to define a local proxy geometry. As each local geometry only has to remain consistent across the small set of neighboring views used to produce a novel view, this makes it possible to represent complex curved reflectors in a 'piecewise' way, helping to maintain sharp reflections.

For per-view geometry, we use a density field approximated by a mixture of ten Gaussians per ray. The Gaussian parameters per ray are encoded via an MLP into a feature, which is then stored within a 2D hash grid per photo (Fig. 2). To calculate soft visibility, we use the Gaussian mixture cumulative distribution function (CDF) along the ray. During rendering, input photo colors are warped backwards from a small set of neighboring views, blended using optimized weights from an MLP, and alpha-composed. This local representation performs well in modeling curved reflectors, producing sharper reflections on objects like cars and glass busts than Ref-NeRF and 3DGS, and producing as sharp results as the global front-facing multi-plane geometry method NeX [Wizadwongsa et al. 2021] without being restricted to front-facing scenes.

In summary, our main technical contributions are as follows.

- A per-view Gaussian density mixture representation and image-based rendering approach that is well-suited to modeling high-frequency reflections for curved and transparent objects.
- An end-to-end optimization scheme with photometric and consistency losses to encourage coherence across per-view proxies, and a sparse voxel grid sampling for efficiency.

Compared with other state-of-the-art neural- and image-based rendering methods, our method can produce sharper results with high-fidelity view-dependent appearance (Fig. 1).

2 RELATED WORK

Warping Image-based Rendering. Unstructured lumigraph rendering [Buehler et al. 2001] warps and interpolates a collection of input images through a proxy geometry. Many methods rely on global geometry reconstructed from captured images [Chaurasia et al. 2011; Goesele et al. 2010; Ortiz-Cayon et al. 2015]. Global geometry can constitute depth images, visual and opacity hulls for pixel visibility, and 3D meshes for view-dependent texturing and surface light fields [Chaurasia et al. 2013; Debevec et al. 1996; Fitzgibbon et al. 2005; Matusik et al. 2000, 2002; Wood et al. 2000]. Later methods use per-view information to improve rendering quality. For instance, Chaurasia et al. [2013] use super-pixels as constraints to derive per-pixel depth, which significantly mitigates image warping artifacts along occlusion edges. Hedman et al. [2016] reconstruct a global geometry and refine the depth map for each view to align edges between the depth channel and the RGB channels. The resulting per-view meshes effectively handle large occlusions and motion parallax. Subsequently, the DeepBlending method [Hedman et al. 2018] integrates two distinct multi-view stereo (MVS) reconstructions for per-view depth refinement. To reduce ghosting, a deep neural network blends images warped with per-view meshes.Wang et al. [2021] also combine image-based rendering method with neural networks and propose a generic view interpolation method.While these methods can reproduce view-dependent effects to some extent, they struggle with specular effects due to their reliance on a proxy geometry with only a single surface—this often fails to represent reflections well without considering reflected rays.

Layered Reflections. We might also try to separate the input images into separate layers containing reflections [Kopf et al. 2013; Szeliski et al. 2000]. Sinha et al. [2012] estimate the foreground and reflected depth to handle planar reflections. Xu et al. [2021] explicitly reconstructed a two-layer geometry along with diffuse and reflected images. Rodriguez et al. [2020] estimate curved reflector geometry for car windows with a two-layer representation—background and car window—using reflective flows. These methods advance our ability to model reflections in IBR by introducing a reflection layer.

Layered Geometry. These representations help to handle complex occlusions and appearance. Layered depth images (LDI) [Shade

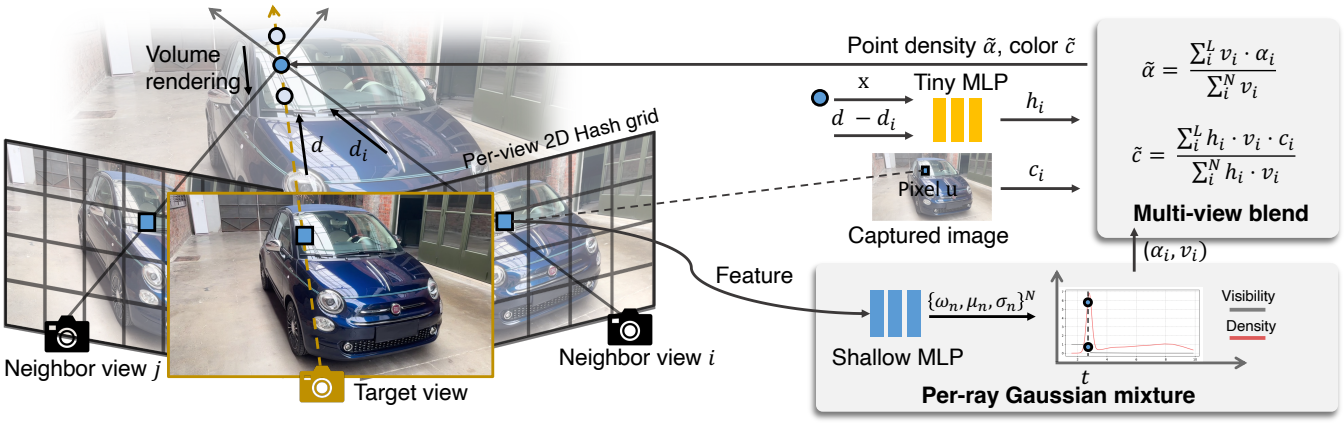

Fig. 2. **Our proposed representation.** *Left*: For each view, the local multi-layer geometry is represented as a per-view density field, with each pixel associated with a ray-based Gaussian mixture parameterized as $\{\mu_n, \sigma_n, \omega_n\}$. All Gaussian mixtures with their parameters in each view are encoded in a 2D hash grid. *Right*: During rendering, for each ray in the target view, we sample a set of points and generate each point's density $\tilde{\alpha}^k$ and color $\tilde{\mathbf{c}}^k$ based on backward warping and occlusion-aware neural weighted blending. Then, the final color of each ray is obtained using alpha composition, which is used in the rendering loss to end-to-end optimize the parameters of all 2D hash features and the MLPs.

et al. 1998] store scene geometry within a projective volume at a specific viewpoint. Penner et al. [2017] extend this concept by constructing projective volumes with additional depth uncertainty for captured images, leading to higher quality view synthesis at occlusion edges. Hedman et al. [2017] use two-layer color-and-depth panoramas to produce perspective views near captured viewpoints with motion parallax effects. Layered representations can also be predicted using deep neural networks, such as end-to-end deep stereo for unstructured view interpolation [Flynn et al. 2016], and deep view synthesis based on multiplane image (MPI) techniques [Li and Khademi Kalantari 2020; Mildenhall et al. 2019; Srinivasan et al. 2019; Wizadwongsa et al. 2021; Xu et al. 2019; Zhou et al. 2018]. LLFF [Mildenhall et al. 2019] and NeX360 [Phongthawee et al. 2022] extend a single MPI to multiple MPIs, where novel views are rendered by blending adjacent MPIs.

Our method connects to layered representations as we use a fixed number of Gaussians per ray. This is related to mixture density distributions in stereo matching [Tosi et al. 2021]. However, unlike an MPI that shares the same depth sampling for all pixels in each plane, our method employs individual depth sampling for each ray.

Global scenes. Many methods create a global representation via reconstruction. Neural radiance fields (NeRF) encode scene density and appearance compactly within a multi-layer perceptron (MLP) neural network. This is a volume rendered to produce an image. Volume rendering has been accelerated through hash grids and direct voxel storage [Fridovich-Keil et al. 2022; Müller et al. 2022], scaled through tiling [Wu et al. 2023, 2022], and extended to dynamic scenes [Pumarola et al. 2021]. Our approach also uses 2D hash grids to encode the local Gaussian mixtures.

Some neural methods are designed for complex appearance [Ma et al. 2024; Verbin et al. 2024; Wu et al. 2024]. Ref-NeRF [Verbin et al. 2022] optimizes a spatial MLP to predict diffuse colors and surface normals and then produces specular reflections via normal-reflected rays and a directional MLP. Other inverse rendering representations incorporate surface details, normals, lighting, albedo, and bidirectional reflectance distribution functions (BRDFs) [Hasselgren et al. 2022; Laine et al. 2020; Liu et al. 2019; Munkberg et al. 2022; Yao et al. 2022; Zhang et al. 2021a,b]. These approaches can be susceptible to over-fitting and are sensitive to initialization and regularization.

Points or primitives have also been studied [Kopanas et al. 2021; Lassner and Zollhöfer 2021; Yifan et al. 2019]. Kopanas et al. [2022] separate reflections among a point cloud using a neural warp field. 3D Gaussian splatting (3DGS) [Kerbl et al. 2023] uses anisotropic Gaussians with color and density as scene primitives, achieving faster rendering speed compared to NeRF-based approaches.

Our method also uses Gaussians to represent the density. However, instead of optimizing a global representation of color and density, we optimize local per-ray density only and use input views for color rendering. This scheme better renders a high-frequency view-dependent appearance. Further, since our Gaussian representation is 1D along each ray, it is challenging to determine the coverage area to be splatted onto other views, so we ray cast instead of splat.

3 METHOD

3.1 Local Gaussian Density Mixture Representation

Our input is a set of M input images $\{I_i\}_{i=1}^M \in \mathcal{I}$ with corresponding camera poses $\{P_i\}_{i=1}^M$. Our objective is to reconstruct a local density field $\{\mathcal{D}_i\}_{i=1}^M$ that, once rendered using IBR in neighboring views, will reproduce the input images. We will take a volume rendering approach. The density distributions in the fields $\mathcal{D}$ are parametrically represented by a Gaussian mixture per ray with $N = 10$ kernels.

Each pixel $\mathbf{u}$ in the image I corresponds to a ray with direction $\mathbf{d}$. Given a camera origin $\mathbf{o}$, a point $\mathbf{x} = \mathbf{o} + t\mathbf{d}$ exists along the ray at a distance t. The volume density at the point $\alpha(\mathbf{x}) = \alpha(\mathbf{u}, t)$ is a

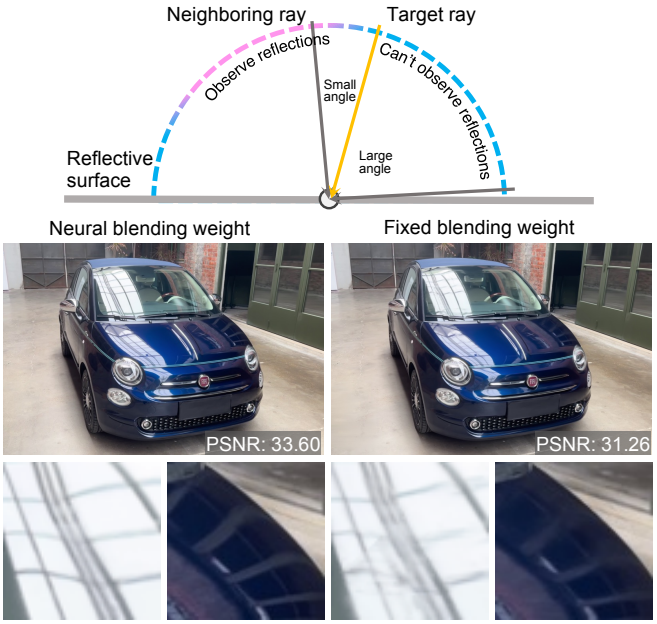

Fig. 3. **Neural blending weight benefit.** *Top*: The dashed pink curve denotes the range of viewing angles in which a reflection is visible. Given a target ray (yellow arrow) that falls outside this range, a neighboring ray with a smaller angle may not capture the similar reflection component. However, using a fixed blending weight function, this neighboring ray will still be assigned a larger weight. *Bottom*: Rendering results using neural blending weights versus a fixed blending weight function.

weighted sum of Gaussians:

$$\alpha(\mathbf{u}, t) = \sum_{n=1}^{N} \omega_n(\mathbf{u})\, g(t; \mu_n(\mathbf{u}), \sigma_n(\mathbf{u})), \tag{1}$$

where each Gaussian $g(\cdot)$ has an associated mean $\mu_n(\mathbf{u})$, standard deviation $\sigma_n(\mathbf{u})$, and weight $\omega_n(\mathbf{u})$.

The soft visibility v or transmittance for each point $\mathbf{x}$ can be calculated analytically from the Gaussian mixture parameterization:

$$\begin{aligned} v(\mathbf{u}, t) &= \exp\left(-\int_s^t \alpha\,(\mathbf{u}, \delta)\, d\delta\right) \\ &= \exp\left(-\sum_{n=1}^{N} \omega_n(\mathbf{u})\,(G(t; \mu_n(\mathbf{u}), \sigma_n(\mathbf{u})) - G(s; \mu_n(\mathbf{u}), \sigma_n(\mathbf{u})))\right) \end{aligned} \tag{2}$$

where s denotes the location of the near plane and $G(t; \mu, \sigma)$ is the cumulative distribution function (CDF) of the Gaussian function:

$$G(t; \mu, \sigma) = \frac{1}{2}\mathrm{erf}\left(\frac{t-\mu}{\sigma\sqrt{2}}\right) + \frac{1}{2}, \tag{3}$$

where $\mathrm{erf}(\cdot)$ is the error function.

3.2 Warping and Fusing Volumes

Next, we introduce a differentiable volume rendering procedure to create novel views from local Gaussian mixtures. The novel target view is formed in three steps via a ray sampling approach, using information from a set of L input views that are neighbors to the target view. First, we reproject local density into the target view's frame via *backward* warping. Then, we merge local densities while considering occlusion by *fusion*. Finally, we *alpha composite* input colors via a neural blending weight. Note that we will use symbols without subscripts to refer to properties of the target view.

Backward warping. For each pixel coordinate $\mathbf{u}$ in the target view, we sample a set of world-space points $\{\mathbf{x}^k\}_{k=1}^K$ along the ray, transform each into the camera space of each neighboring view i, then project the point to pixel coordinates to produce $\mathbf{u}'_i$. With this, we can obtain a color $\mathbf{c}_i^k$, density α_i^k, and visibility v_i^k:

$$\mathbf{u}'_i = \pi(\mathbf{x}^k; P_i),\ \mathbf{c}_i^k = I_i(\mathbf{u}'_i), \tag{4}$$

$$\alpha_i^k = \alpha_i(\mathbf{u}'_i, ||\mathbf{x}^k - \mathbf{o}_i||),\ v_i^k = v_i(\mathbf{u}'_i, ||\mathbf{x}^k - \mathbf{o}_i||), \tag{5}$$

where $\pi_i(\cdot)$ denotes the projection operation from a point in world space to the pixel space of the i-th neighboring input view. We calculate density and visibility using Eq. 1 and 2, respectively.

Fusion. We fuse the warped densities and colors by considering visibility and using an optimized blending weight. As we expect density to represent local and view-independent geometry, we consider only visibility when fusing:

$$\tilde{\alpha}^k = \frac{\sum_i v_i^k \cdot \alpha_i^k}{\sum_i v_i^k}. \tag{6}$$

Color will still contain view-dependent information even with local warping. As a result, we fuse the multi-view colors using a neural blending weight h_i^k:

$$\tilde{\mathbf{c}}^k = \frac{\sum_i h_i^k \cdot v_i^k \cdot c_i^k}{\sum_i h_i^k \cdot v_i^k}. \tag{7}$$

h_i^k is encoded in a small MLP:

$$h_i^k = \Phi(\mathbf{x}^k, \mathbf{d}^k - \mathbf{d}_i^k; \theta), \tag{8}$$

where $\mathbf{d}^k$ and $\mathbf{d}_i^k$ indicate the ray direction for the point $\mathbf{x}^k$ for the target and neighboring views, and θ represents the MLP parameters. We use the point position and relative ray direction as inputs to allow view dependence.

In comparison to a fixed blending weight function (ULR [Buehler et al. 2001], InsideOut [Hedman et al. 2016]), our neural blending weights are optimized end-to-end and can better capture high frequency reflection (Fig. 3). Compared to weights predicted by a pre-trained CNN (DeepBlending [Hedman et al. 2018]), our neural blending weights use a compact MLP.

Alpha composition. Finally, we accumulate a fused color $\tilde{\mathbf{c}}^k$ along the ray by using the fused densities $\tilde{\alpha}^k$. This step obtains the rendered color for each pixel:

$$\tilde{\mathbf{c}} = \sum_k \left(w^k \cdot \tilde{\mathbf{c}}^k\right), \tag{9}$$

$$w^k = \exp(-\sum_{j=1}^{k-1} \tilde{\alpha}^j \cdot \delta^j) \cdot (1 - \exp(-\tilde{\alpha}^k \cdot \delta^k), \tag{10}$$

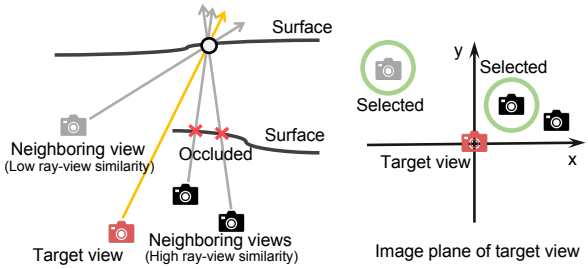

Fig. 4. **View selection.** *Left*: Selecting views based only on ray-view similarity can result in certain points not being trained with enough observations. Rendering artifacts may appear for the surface point invisible to two of the selected neighboring views. *Right*: This issue can be fixed by selecting views based on their projected 2D positions with respect to the target view, ensuring they are uniformly distributed across all four quadrants when projected onto the image space of the target view.

where w^k represents the alpha-blending weight for the point $\mathbf{x}_k$ in the target view.

Neighbor view selection. Even with visibility-aware and view-dependent fusion, picking a set of good neighbor views L is critical to achieving high quality results. This selection is guided by an accumulated cosine similarity per ray that considers the angle between the target and neighbor view rays:

$$\mathrm{S}_i = \sum_k \mathrm{S}_i^k = \sum_k \frac{(\mathbf{o} - \mathbf{x}^k) \cdot (\mathbf{o}_i - \mathbf{x}^k)}{||\mathbf{o} - \mathbf{x}^k|| \cdot ||\mathbf{o}_i - \mathbf{x}^k||}. \tag{11}$$

S_i^k is set to 0 if the projected pixel falls outside the viewport.

Given ray-view similarities $\{\mathrm{S}_i\}$ for all neighboring views, next we select the L views. Naïvely selecting views with the highest similarity values may select uninformative 'clumps' of cameras (Fig. 4 *left*). Instead, we stratify selection (Fig. 4 *right*): We project the centers of all neighboring cameras onto the image plane of the target view, which separates neighboring cameras centers into four quadrants. Then, we rank view similarity within each quadrant and iteratively select neighboring views with the highest similarity values from each quadrant in turn. This approach tends to produce more balanced neighbors. For instance, when observing an occlusion edge, neighbor cameras in one direction will always fail to see a point beyond the edge. Our strategy avoids selecting only those cameras even if they are nearby.

3.3 Optimization

To optimize the local density fields and neural weights, we define a loss $\mathcal{L}$ with two components: an L2 reconstruction loss $\mathcal{L}_\mathrm{r}$ and a consistency loss $\mathcal{L}_\mathrm{c}$:

$$\mathcal{L} = \mathcal{L}_\mathrm{r} + 0.01\mathcal{L}_\mathrm{c}, \tag{12}$$

Reconstruction loss. To optimize geometry that is local to each view, we must enforce constraints between views. Hence, during optimization, each input view is treated as a target view to be reconstructed. For this, we minimize the difference between rendered pixel colors $\tilde{\mathbf{c}}$ and their matched input photograph pixel colors $\mathbf{c}$:

$$\mathcal{L}_\mathrm{r} = ||\tilde{\mathbf{c}} - \mathbf{c}||_2^2. \tag{13}$$

Consistency loss. Even with a reconstruction loss, the density field local to an input view may be inconsistent with the fused density field produced when the input view is treated as a target view. We add a consistency loss by minimizing the KL divergence between alpha-blending weights of K sampled points along rays:

$$\mathcal{L}_\mathrm{c} = \sum_k \tilde{w}^k \cdot \log(\frac{\tilde{w}^k}{w^k}), \tag{14}$$

where $\tilde{w}^k$ is the alpha-blending weight obtained from the fused density field of the target view, and w^k is the alpha-blending weight derived from the (same) input view's local density field.

Although it is only a soft constraint, the consistency loss plays an important role in improving the view consistency of the rendering results, as adding it achieves more accurate geometry in general (Fig. 8). We can consider the loss' effect in two ways: 1) scene areas that rely on density alignment, like occlusion edges, are encouraged to be similar; and 2) scene areas that can be reproduced using a variety of different geometries (say, low-textured regions) are encouraged to be similar.

4 IMPLEMENTATION DETAILS

Hash grid and network architecture. Storing the Gaussian mixture mean, standard deviation, and weight parameters requires $W \times H \times N \times 3$ floats for each $W \times H$ image, which may exceed available GPU memory during optimization. To overcome this, we reduce memory use by representing the Gaussian parameters compactly. The approach uses a hash table of features unique to each image, followed by a shallow MLP shared across all features (Fig. 2) [Müller et al. 2022]. The hash table features are collectively optimized such that the Gaussian mixtures can reproduce the scene. This allows similar ray density distributions to share MLP capacity through the embedding space while allowing the image-space location of those distributions to move.

Each 2D hash grid has 16 levels, where the coarsest level resolution is 16×16 (e.g., in a 640×480 image, each cell covers 40×30 pixels at the coarsest level due its 16×16 subdivision) and the highest resolution matches that of the input view. The feature size at each level is two, and the hash table size for each 2D hash grid is 2^{16}.

We use three MLPs. The first MLP decodes features from the hash grid to produce the Gaussian mixture parameters. The second MLP decodes features from the hash grid to produce colors. Both of these MLPs have two 64-neuron layers. The third MLP produces the neural blending weight and comprises two sub-modules. The first sub-module encodes 3D points into 16-dimension features. These features are concatenated with ray directions and input to the second sub-module. Both submodules have two 32-neuron layers. Every MLP is Gaussian activated [Ramasinghe and Lucey 2022] with a variance of 0.01.

Baking and rendering. To accelerate rendering at the cost of memory, we can precompute and store the optimized Gaussian mixture parameters $\{\mu_n, \sigma_n, \omega_n\}$ for each view as a 2D grid. This eliminates the computational overhead of hashing and executing the MLP to

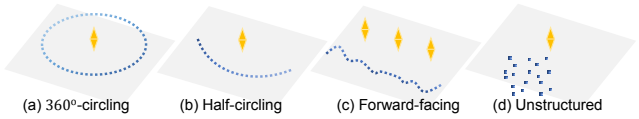

Fig. 5. **Four types of camera trajectories in the LGDM dataset.**

recover these parameters. For the neural blending weights, the MLP is small enough to be stored in CUDA shared memory. For rendering, our method samples 192 points for each ray in the target view and blends density and colors for each point across 8 neighboring views.

Ray sampling strategy. To maintain balanced gradient scales during each optimization iteration, we adopt a strategy of randomly sampling an equal number of pixels for each view.

Point sampling and occupancy grid. We use an occupancy grid to speed up volume point sampling. For every sample point, if the visibility-aware weight w (Eq. 10) exceeds 0.01, the corresponding voxel covering this point is occupied. When processing each target ray, we begin by sampling 64 points in disparity space. Subsequently, we uniformly sample 128 points only within the occupied voxels.

Further, we adopt a strategy of gradually subdividing our occupied grid to increase its resolution [Liu et al. 2020; Wu et al. 2023]. This subdivision is performed every 1,000 iterations. We start with an initial resolution of 8^3 and progressively increase it until reaching a resolution of 512^3.

Viewport extension. In certain scenes, some points are only observed from the target view and so will warp outside all neighboring views. During optimization, since each input view is treated as a target view, these points do not have information from neighboring views to correctly minimize the reconstruction loss, resulting in floating geometries. To address this issue, we extend each input view by 50 pixels on each side. The 2D hash feature grid is also extended accordingly. For the extended pixels, as they do not have captured colors, we reuse the 2D hash grid features to generate color for the extended pixels, using an additional color-MLP shared for all the images. The color for the extended pixels is optimized with respect to the target view. After optimization, the extended feature grids and the color-MLP are discarded.

Hyperparameters configuration. Throughout all of our experiments, we optimize for 60k iterations, with each batch consisting of 2,048 rays. We use the *Adam* [Kingma and Ba 2014] optimizer with a learning rate that decays from $1e^{-3}$ to $1e^{-4}$. The training time for each scene is approximately 4-5 hours on a single Nvidia V100 GPU.

5 EXPERIMENTS

Baselines. We compare to NeX [Wizadwongsa et al. 2021] which uses a single multi-plane image (MPI) with a neural view-dependent appearance basis; LLFF [Mildenhall et al. 2019] which fuses local multi-plane image representations; Instant-NGP (INGP) [Müller et al. 2022] which models scenes as global radiance fields; 3DGS [Kerbl et al. 2023] which uses primitives to describe a radiance field; Ref-NeRF [Verbin et al. 2022] and NeuralCatacaustics (NPC) [Kopanas et al. 2022], which are both designed to handle curved reflections.

Table 1. **Quantitative comparisons on our new LGDM dataset.** Best results are highlighted as 1st, 2nd and 3rd. The type of camera trajectory for each captured scene is visualized in Fig. 5.

Scene	Metric	LLFF	NPC	NeX	Ref-NeRF	INGP	3DGS	Ours
Blue Car	PSNR↑	25.64	25.14	28.77	29.00	27.11	31.63	32.82
67 views	SSIM↑	0.903	0.913	0.930	0.915	0.907	0.965	0.974
Forward-facing	LPIPS↓	0.111	0.229	0.215	0.323	0.299	0.153	0.068
Red Car	PSNR↑	26.42	22.62	N/A	28.11	27.00	28.52	31.88
128 views	SSIM↑	0.905	0.859	N/A	0.894	0.913	0.950	0.964
Half-circling	LPIPS↓	0.137	0.292	N/A	0.291	0.198	0.141	0.091
Natatorium	PSNR↑	23.98	23.88	25.01	25.82	25.67	27.43	31.22
144 views	SSIM↑	0.851	0.867	0.846	0.862	0.879	0.929	0.960
Forward-facing	LPIPS↓	0.119	0.229	0.287	0.277	0.219	0.136	0.071
Glass Bust	PSNR↑	26.28	20.89	N/A	27.92	25.73	29.42	33.48
194 views	SSIM↑	0.883	0.821	N/A	0.894	0.871	0.954	0.971
Half-circling	LPIPS↓	0.165	0.389	N/A	0.327	0.373	0.125	0.087
Skyscraper	PSNR↑	20.67	20.21	25.89	24.26	21.95	27.83	30.66
132 views	SSIM↑	0.786	0.844	0.880	0.827	0.792	0.942	0.961
Forward-facing	LPIPS↓	0.137	0.253	0.233	0.339	0.332	0.099	0.073
Mall	PSNR↑	24.83	26.12	30.29	28.15	28.07	24.79	32.53
112 views	SSIM↑	0.879	0.922	0.948	0.907	0.916	0.907	0.969
Forward-facing	LPIPS↓	0.100	0.203	0.129	0.240	0.212	0.198	0.073
Bull	PSNR↑	25.88	23.02	25.95	25.38	25.25	26.63	28.90
233 views	SSIM↑	0.873	0.832	0.873	0.840	0.862	0.909	0.932
Unstructured	LPIPS↓	0.186	0.298	0.330	0.408	0.298	0.211	0.148
Sculpture	PSNR↑	21.81	19.33	N/A	24.12	22.69	25.28	27.01
150 views	SSIM↑	0.783	0.750	N/A	0.810	0.801	0.900	0.912
360°-circling	LPIPS↓	0.263	0.433	N/A	0.419	0.422	0.203	0.189
	PSNR↑	24.44	22.65	N/A	26.60	25.43	27.69	31.06
Mean	SSIM↑	0.858	0.851	N/A	0.868	0.868	0.932	0.955
	LPIPS↓	0.152	0.290	N/A	0.328	0.294	0.158	0.100

Table 2. **Quantitative comparisons on the Shiny dataset.**

Method	Metric	CD	Tools	Crest	Seasoning	Food	Giants	Lab	Pasta	Mean
NeX	PSNR↑	**31.43**	**28.16**	21.23	**28.60**	23.68	**26.00**	30.43	**22.07**	**26.45**
	SSIM↑	0.958	**0.953**	**0.757**	**0.928**	0.832	**0.898**	0.949	**0.844**	**0.890**
	LPIPS↓	0.129	0.151	0.162	**0.168**	0.203	**0.147**	0.146	**0.211**	0.165
Ours	PSNR↑	31.34	27.95	**21.40**	28.55	**24.22**	24.12	**32.40**	21.44	26.43
	SSIM↑	**0.981**	0.952	0.749	0.926	**0.849**	0.845	**0.981**	0.832	**0.890**
	LPIPS↓	**0.083**	**0.137**	**0.148**	0.169	**0.180**	0.177	**0.080**	0.219	**0.150**

We use the original implementations and hyperparameters provided by the respective authors.

5.1 Evaluation on LGDM Data

To evaluate reflections on wide motion scenes, we capture a new dataset '*LGDM*' with 8 scenes showing prominent or complex reflections, including multi-layer planar reflections (*Mall*), large glass surfaces (*Skyscraper*, *Natatorium*), curved surface reflections (*Blue Car*, *Red Car*, *Sculpture*), refraction (*Glass Bust*, *Bull*). This dataset is captured with 67–233 images, resized from 4K (3840 × 2160) to 1K resolution for training and evaluation, and the types of camera trajectory used in the capturing are illustrated in Fig. 5. We select a subset of the images as hold-out views (12.5%) for evaluation. The camera poses are computed using *COLMAP* [Schönberger and Frahm 2016]. *Skyscraper* and *Sculpture* were captured with a DJI MINI 4 PRO drone, and the remaining scenes were captured with an iPhone 14 Pro Max.

Quantitatively, our method outperforms all compared methods (Table 1), increasing PSNR by 2.64 dB on average. Among these results, NeX [Wizadwongsa et al. 2021] exhibits a significant decrease in PSNR when confronted with larger motion parallax due to its reliance on a single MPI. As *Sculpture* is a 360° scene, and the camera

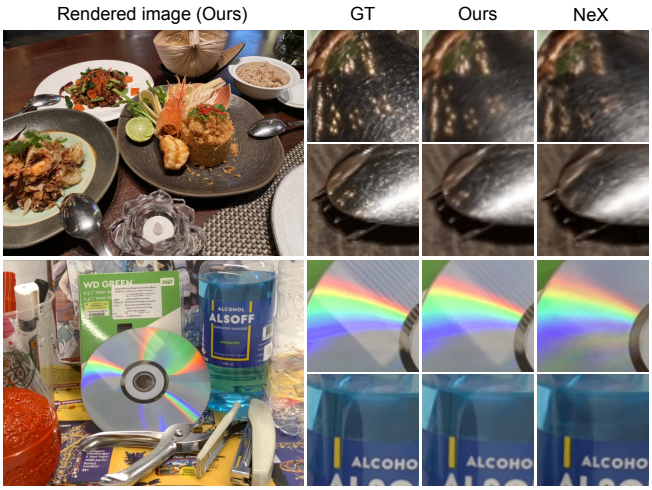

Fig. 6. **Comparisons on Shiny dataset.** Compared with NeX [Wizadwongsa et al. 2021], our local representation can faithfully reproduce the highlights and reflections on the stainless steel (top row), as well as the reflected contents on the CD and the bottle (bottom row).

circles around the *Red Car* and *Glass Bust* from 120° to 180°, NeX's single MPI is not suitable for these scenarios and so we mark them as N/A. 3DGS [Kerbl et al. 2023] suffers from a brittle optimization in challenging scenes. For instance, in *Mall*, where the content on the TV is dynamic and the texture and reflections on the floor are high frequency, we see floating geometries that lead to a large drop in PSNR (Fig. 11). To enable direct comparison between rendered and captured views, we intentionally design the camera trajectory to be close to the captured views via interpolating between selected keyframe camera poses in this demo for the *Red Car* scene. A video comparison (1:16–1:23) shows that our method outperforms SOTA methods in reproducing fine details such as the reflections on the polished car surface and the flare on the car window.

While LLFF [Mildenhall et al. 2019] also uses a local representation and generates visually appealing results, it suffers from minor 'pixel shifting' caused by inaccurate geometry. As reflection-specific methods, Ref-NeRF [Verbin et al. 2022] and NPC [Kopanas et al. 2022] still struggle with large curved reflections, such as the glass in *Natatorium* (Fig. 11). These methods also encounter difficulties in handling transparent objects, as seen in the *Glass Bust* scene.

5.2 Evaluation on Shiny Dataset

The publicly-available Shiny dataset from NeX [Wizadwongsa et al. 2021]) includes complex reflections from CDs and refractions through water bottles. The motion parallax and depth of field in this dataset are relatively smaller than in our LGDM dataset. Both NeX and our method produce similar quantitative results (Tab. 2); however, qualitative results reveal that our method reproduces sharper reflections with more details (Fig. 6). For example, while the CD scene show similar PSNR for both methods, our approach reproduces the linear striped pattern on the CD itself whereas NeX does not. Average LPIPS decreases from 0.165 to 0.150, suggesting this improved perceptual quality over NeX [Wizadwongsa et al. 2021].

Table 3. **Quantitative comparisons on the Real Forward-Facing dataset.** Best results are highlighted as 1st, 2nd and 3rd. Please refer to the supplementary material for the metrics on each scene.

Scene	Metric	LLFF	NeRF	NeX	INGP	3DGS	Ours
Mean	PSNR↑	24.41	26.76	27.26	24.84	24.86	27.18
	SSIM↑	0.863	0.883	0.904	0.855	0.876	0.905
	LPIPS↓	0.211	0.246	0.178	0.262	0.197	0.166

Table 4. **Ablation on losses.** Without the consistency loss, the PSNR decreases for both scenes in the RFF dataset and our LGDM dataset.

	Metric	*Fern*	*Flower*	*T-Rex*	*Natatorium*	*Glass Bust*
w/o $\mathcal{L}_c$	PSNR↑	24.55	28.53	26.71	31.16	33.18
	SSIM↑	0.853	0.928	0.928	0.960	0.971
	LPIPS↓	0.231	0.141	0.199	0.069	0.084
Ours	PSNR↑	25.58	29.15	27.86	31.22	33.48
	SSIM↑	0.880	0.934	0.043	0.960	0.971
	LPIPS↓	0.193	0.130	0.165	0.071	0.084

5.3 Evaluation on Real Forward-facing dataset

On the publicly-available real forward-facing (RFF) dataset from NeRF [Mildenhall et al. 2020], our approach show competitive performance with NeX [Wizadwongsa et al. 2021], where both show higher average metrics than other methods (Tab. 3). INGP [Müller et al. 2022] converges quickly but struggles to consistently produce details. 3DGS [Kerbl et al. 2023] can generate sharp results for certain scene areas but exhibits significant artifacts, such as floating geometries.

5.4 Ablations

Consistency loss. We select *Fern*, *Flower*, and *T-rex* from the RFF dataset, as well as *Natatorium* and *Glass Bust* from our LGDM dataset to evaluate the impact of the consistency loss (Tab. 4). Removing the consistency loss $\mathcal{L}_c$ significantly reduces PSNR. Qualitatively, removing it results in geometry exhibiting noise and missing details, leading to noticeable artifacts during rendering (Fig. 8).

Number of Gaussians and selected views. On the *Fern* data from the RFF dataset, reducing the number of Gaussians can result in significant missing geometry (Fig. 9). Similarly, employing fewer selected views for warping and blending can present challenges in accurately locating geometric surfaces, potentially leading to undesirable artifacts such as blurring and ghosting in the final results. Through experimentation, we show that using 10 Gaussians and eight selected views for each ray is a reasonable balance between computational efficiency and reconstruction quality.

Captured image baseline. In ray interpolation-based IBR, the baseline between captured images is a key factor that influences rendering results. We decrease sampling by 1/2, 1/4, and 1/8 of the total number of training views (Fig. 10 and Tab. 5). As sparsity increases, fewer input views indeed reduces rendering quality; at 1/8th of the input views, we see noticeable artifacts in the rendered images.

Table 5. **Ablation on the density of the captured views.** 1/2, 1/4, and 1/8 represent the proportion of the full training set. We uniformly sample views from full set as training views.

Scene	Metric	Full Set	1/2	1/4	1/8
Glass Bust	PSNR↑	33.48	31.70	29.25	27.07
	SSIM↑	0.971	0.962	0.940	0.910
	LPIPS↓	0.084	0.101	0.130	0.166
Natatorium	PSNR↑	31.22	30.33	28.96	27.00
	SSIM↑	0.960	0.952	0.935	0.907
	LPIPS↓	0.071	0.085	0.104	0.140

6 CONCLUSION

We have shown that a per-view Gaussian density mixture with image-based rendering can be end-to-end optimized to achieve high-frequency reflections for curved and transparent objects.

Limitations and future work. As an IBR method, our results can still show some visual 'snapping' on curved reflections as the target view moves between different sets of neighbors. In these cases, there is a visual trade-off vs. global scene methods between such snapping in our case and blurring in the case of Ref-NeRF and 3DGS.

The approach is not yet constructed for fast rendering as it uses a volume sampling method. Each 952×535 view takes around 270 ms to render on an NVIDIA RTX 3090Ti, which is faster than NeRF but is slower than 3DGS. One way to increase rendering speed is to reduce the number of sampled points per ray. A coarse-to-fine strategy per ray may increase speed without reducing quality.

Our method achieves highest quality with dense scene capture, especially for complex reflections or refractions. In diffuse areas, this results in redundant duplication. But, even with the many views in our LGDM dataset, global scene representations like Ref-NeRF or 3DGS still cannot achieve as high a quality of reflections as ours; this discrepancy seems pertinent to investigate in future work.

Another direction for future work is to explore why our method reconstructs view-dependent effects better than other approaches, particularly methods that construct global geometries. We have observed that flexible local geometries and direct color blending are factors in achieving these results. Developing a rigorous evaluation framework or theory to support these findings would be valuable.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their professional and constructive comments. Weiwei Xu is partially supported by NSFC grant No. 61732016. Jiamin Xu is partially supported by NSFC grant No. 62302134 and ZJNSF grant No. LQ24F020031. James Tompkin is partially supported by the US NSF CNS-2038897 and by CAREER IIS-2144956. Qixing Huang is partially supported by US NSF CAREER IIS-2047677 and by US NSF CAREER IIS-2413161. Yifan Peng is partially supported by Hong Kong University Grants Committee (ECS 27212822, GRF 17208023) and National Natural Science Foundation of China. This paper is supported by Ant Group and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

REFERENCES

C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. 2001. Unstructured lumigraph rendering. In *ACM Trans. Graph.* 425–432.

G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.* 32, 3 (2013), 1–12.

G. Chaurasia, O. Sorkine-Hornung, and G. Drettakis. 2011. Silhouette-Aware Warping for Image-Based Rendering. In *Computer Graphics Forum*, Vol. 30. 1223–1232.

P. E. Debevec, C. J. Taylor, and J. Malik. 1996. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *SIGGRAPH, ACM*. 11–20.

Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. 2008. Floating textures. In *Computer graphics forum*, Vol. 27. Wiley Online Library, 409–418.

Andrew Fitzgibbon, Yonatan Wexler, and Andrew Zisserman. 2005. Image-based rendering using image-based priors. *International Journal of Computer Vision* 63 (2005), 141–151.

John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5515–5524.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5501–5510.

M. Goesele, J. Ackermann, S. Fuhrmann, C. Haubold, R. Klowsky, D. Steedly, and R. Szeliski. 2010. Ambient Point Clouds for View Interpolation. In *SIGGRAPH, ACM*. Article 95, 6 pages.

S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. 1996. The lumigraph. In *SIGGRAPH, ACM*. 43–54.

Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, light & material decomposition from images using monte carlo rendering and denoising. *arXiv preprint arXiv:2206.03380* (2022).

P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf. 2017. Casual 3D Photography. *ACM Trans. Graph.* 36, 6, Article 234 (2017), 15 pages.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.* 37, 6 (2018), 1–15.

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–11.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (2023).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. 2022. Neural point catacaustics for novel-view synthesis of reflections. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–15.

Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. 2021. Point-Based Neural Rendering with Per-View Optimization. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 29–43.

Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. 2013. Image-based rendering in the gradient domain. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–9.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular primitives for high-performance differentiable rendering. *ACM Trans. Graph.* 39, 6 (2020), 1–14.

Christoph Lassner and Michael Zollhöfer. 2021. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1440–1449.

M. Levoy and P. Hanrahan. 1996. Light field rendering. In *SIGGRAPH, ACM*. 31–42.

Qinbo Li and Nima Khademi Kalantari. 2020. Synthesizing Light Field From a Single Image with Variable MPI and Two Network Fusion. *ACM Transactions on Graphics* 39, 6 (12 2020). https://doi.org/10.1145/3414685.3417785

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural sparse voxel fields. *Advances in Neural Information Processing Systems* 33 (2020), 15651–15663.

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Int. Conf. Comput. Vis.* 7708–7717.

Li Ma, Vasu Agrawal, Haithem Turki, Changil Kim, Chen Gao, Pedro Sander, Michael Zollhöfer, and Christian Richardt. 2024. Specnerf: Gaussian directional encoding for specular reflections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21188–21198.

W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. 2000. Image-Based Visual Hulls. In *SIGGRAPH, ACM*. 6 pages.

W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. Mcmillan. 2002. Image-Based 3D Photography Using Opacity Hulls. *ACM Trans. Graph.* 21, 3 (2002),

427–437.
B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. 2019. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.* 38, 4 (2019), 1–14.
Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis.*
Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* 41, 4 (2022), 1–15.
Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting triangular 3d models, materials, and lighting from images. In *IEEE Conf. Comput. Vis. Pattern Recog.* 8280–8290.
R. Ortiz-Cayon, A. Djelouah, and G. Drettakis. 2015. A Bayesian Approach for Selective Image-Based Rendering Using Superpixels. In *2015 International Conference on 3D Vision.* 469–477.
E. Penner and L. Zhang. 2017. Soft 3D reconstruction for view synthesis. *ACM Trans. Graph.* 36, 6 (2017), 1–11.
Pakkapon Phongthawee, Suttisak Wizadwongsa, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. 2022. Nex360: Real-time all-around view synthesis with neural basis expansion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 6 (2022), 7611–7624.
Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 10318–10327.
Sameera Ramasinghe and Simon Lucey. 2022. Beyond periodicity: towards a unifying framework for activations in coordinate-MLPs. In *Eur. Conf. Comput. Vis.* 142–158.
S. Rodriguez, S. Prakash, P. Hedman, and G. Drettakis. 2020. Image-Based Rendering of Cars using Semantic Labels and Approximate Reflection Flow. *Proc. ACM Comput. Graph. Interact.* 3 (2020).
Johannes L Schönberger and Jan-Michael Frahm. 2016. Structure-from-motion revisited. In *IEEE Conf. Comput. Vis. Pattern Recog.* 4104–4113.
J. Shade, S. Gortler, L. He, and R. Szeliski. 1998. Layered depth images. In *SIGGRAPH, ACM.* 231–242.
Sudipta N Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. 2012. Image-based rendering for scenes with reflections. *ACM Trans. Graph.* 31, 4 (2012), 1–10.
P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. 2019. Pushing the boundaries of view extrapolation with multiplane images. In *IEEE Conf. Comput. Vis. Pattern Recog.* 175–184.
Richard Szeliski, Shai Avidan, and Padmanabhan Anandan. 2000. Layer extraction from multiple images containing reflections and transparency. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, Vol. 1. IEEE, 246–253.
Fabio Tosi, Yiyi Liao, Carolin Schmitt, and Andreas Geiger. 2021. Smd-nets: Stereo mixture density networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 8942–8952.
Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. 2022. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5481–5490.
Dor Verbin, Pratul P Srinivasan, Peter Hedman, Ben Mildenhall, Benjamin Attal, Richard Szeliski, and Jonathan T Barron. 2024. NeRF-Casting: Improved View-Dependent Appearance with Consistent Reflections. *arXiv preprint arXiv:2405.14871* (2024).
Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. 2021. Ibrnet: Learning multi-view image-based rendering. In *IEEE Conf. Comput. Vis. Pattern Recog.* 4690–4699.
Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. 2021. Nex: Real-time view synthesis with neural basis expansion. In *IEEE Conf. Comput. Vis. Pattern Recog.* 8534–8543.
Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. 2000. Surface light fields for 3D photography. In *Proc. of SIGGRAPH.* 287–296.
Liwen Wu, Sai Bi, Zexiang Xu, Fujun Luan, Kai Zhang, Iliyan Georgiev, Kalyan Sunkavalli, and Ravi Ramamoorthi. 2024. Neural Directional Encoding for Efficient and Accurate View-Dependent Appearance Modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 21157–21166.
Xiuchao Wu, Jiamin Xu, Xin Zhang, Hujun Bao, Qixing Huang, Yujun Shen, James Tompkin, and Weiwei Xu. 2023. ScaNeRF: Scalable Bundle-Adjusting Neural Radiance Fields for Large-Scale Scene Rendering. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–18.
Xiuchao Wu, Jiamin Xu, Zihan Zhu, Hujun Bao, Qixing Huang, James Tompkin, and Weiwei Xu. 2022. Scalable neural indoor scene rendering. *ACM Trans. Graph.* 41, 4 (2022), 1–16.
Jiamin Xu, Xiuchao Wu, Zihan Zhu, Qixing Huang, Yin Yang, Hujun Bao, and Weiwei Xu. 2021. Scalable image-based indoor scene rendering with reflections. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.
Z. Xu, S. Bi, K. Sunkavalli, S. Hadap, H. Su, and R. Ramamoorthi. 2019. Deep view synthesis from sparse photometric images. *ACM Trans. Graph.* 38, 4 (2019), 1–13.
Yao Yao, Jingyang Zhang, Jingbo Liu, Yihang Qu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. 2022. Neilf: Neural incident light field for physically-based material estimation. In *Eur. Conf. Comput. Vis.* 700–716.
Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. 2019. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.
Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021a. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *IEEE Conf. Comput. Vis. Pattern Recog.* 5453–5462.
Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. 2021b. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.* 40, 6 (2021), 1–18.
T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. 2018. Stereo magnification: learning view synthesis using multiplane images. *ACM Trans. Graph.* 37, 4 (2018), 1–12.

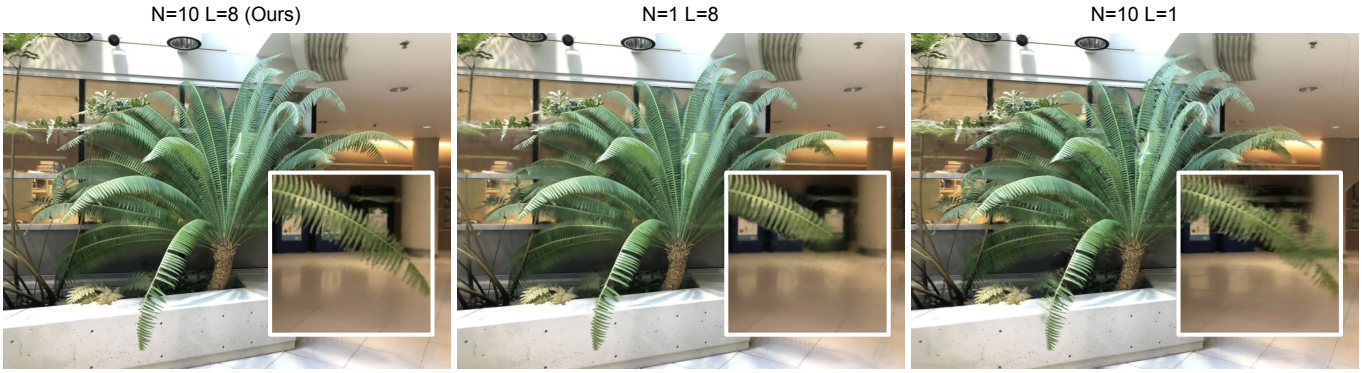

Fig. 7. **Renderings with fewer Gaussians and neighboring views**. A smaller number of Gaussians (N=1) for each ray will result in missing geometry, while a smaller number of neighbor views (L=1) makes it hard to handle occlusions.

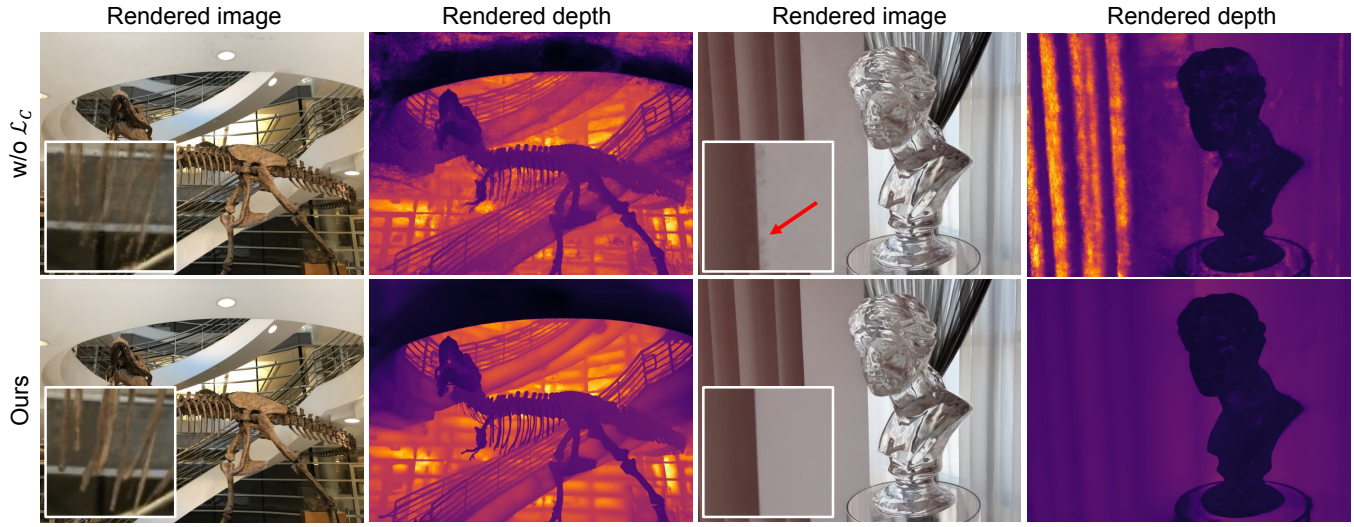

Fig. 8. **Ablation on consistency loss.** As shown in the top row, without the consistency loss, the rendered depth maps exhibit numerous artifacts, resulting in a loss of geometric details (e.g., the ribs of the T-rex) and incorrect depth estimates (e.g., the curtain behind glass bust). The bottom row shows that our consistency loss can help to improve the accuracy of local geometry, leading to fewer artifacts in rendering.

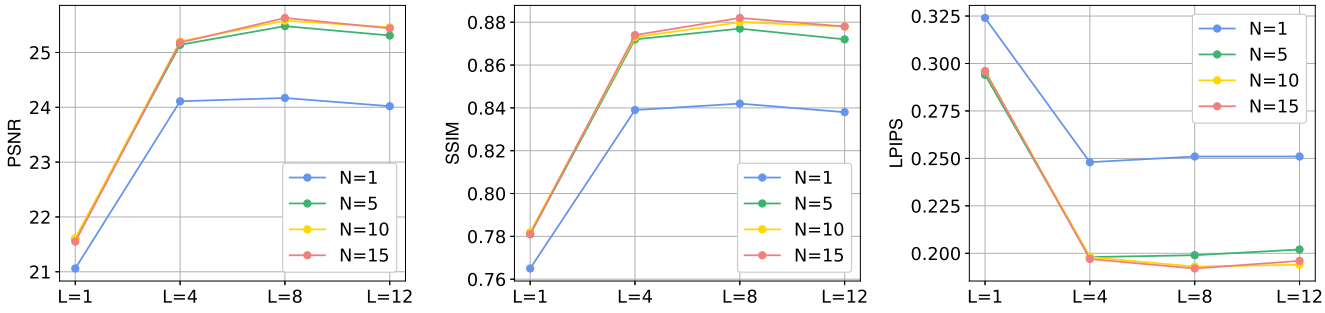

Fig. 9. **Ablation on numbers of Gaussians N and neighboring views L.** The line charts show that the number of Gaussians and neighbor views in our setting (N=10, L=8) is enough for producing high-qulity rendering results. Note that the performance slightly degrades when L reaches 12. This is because a large value of L incorporates more neighboring views; depending on camera sampling, these views may be more distant. As a result, more distant views are likely to induce larger occlusions and more diverse view-dependent appearance, thus making view synthesis more challenging.

Fig. 10. **Varying the baseline of captured views.** In *Natatorium*, while quality stays reasonable with half the views, sharpness decreases with one quarter of the views as the camera baseline increases. At one eighth of the views, artifacts in thin features appear. PSNR metrics are reported for zoom-ins.

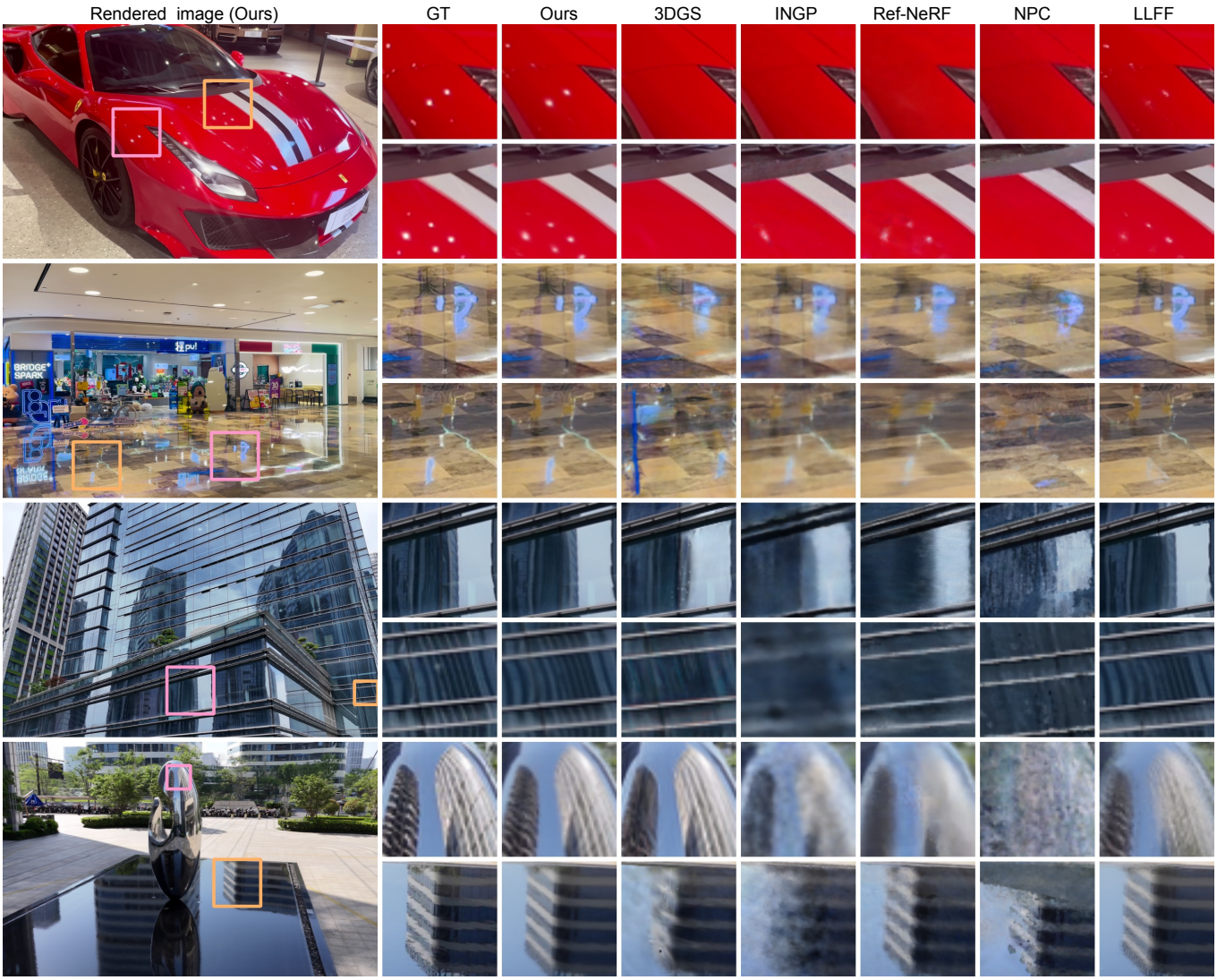

Fig. 11. **Results on our LGDM dataset.** Top to bottom: *Red Car*, *Mall*, *Skyscraper*, *Sculpture*. Overall, compared with 3DGS [Kerbl et al. 2023], INGP [Müller et al. 2022], Ref-NeRF [Verbin et al. 2022], NPC [Kopanas et al. 2022] and LLFF [Mildenhall et al. 2019], our method creates more accurate scene reproductions for reflections. For the *Red Car* scene, we also provide a video comparison in our accompanying video, alongside the captured video serving as ground truth.