VastGaussian: Vast 3D Gaussians for Large Scene Reconstruction

Jiaqi Lin[1*] Zhihao Li[2*] Xiao Tang[2] Jianzhuang Liu[3] Shiyong Liu[2] Jiayue Liu[1]
Yangdi Lu[2] Xiaofei Wu[2] Songcen Xu[2] Youliang Yan[2] Wenming Yang[1†]
[1]Tsinghua University [2]Huawei Noah's Ark Lab [3]Chinese Academy of Sciences
[*] Equal contribution [†] Corresponding author

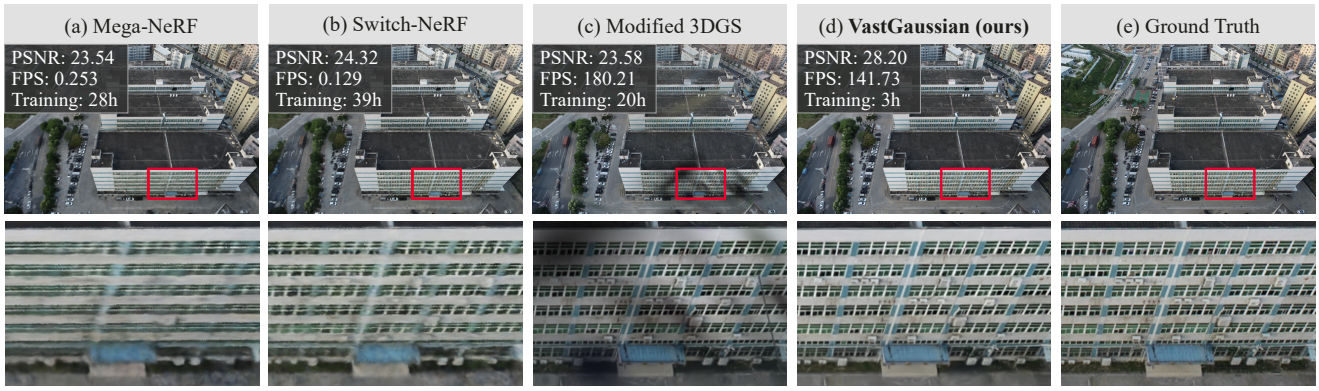

Figure 1. Renderings of three state-of-the-art methods and our VastGaussian from the *Residence* scene in the UrbanScene3D dataset [26]. (a, b) Mega-NeRF [44] and Switch-NeRF [61] produce blurry results with slow rendering speeds. (c) We modify 3D Gaussian Splatting (3DGS) [21] so that it can be optimized for enough iterations on a 32 GB GPU. The rendered image is much sharper, but with a lot of floaters. (d) Our VastGaussian achieves higher quality and much faster rendering than state-of-the-art methods in large scene reconstruction, with much shorter training time.

Abstract

Existing NeRF-based methods for large scene reconstruction often have limitations in visual quality and rendering speed. While the recent 3D Gaussian Splatting works well on small-scale and object-centric scenes, scaling it up to large scenes poses challenges due to limited video memory, long optimization time, and noticeable appearance variations. To address these challenges, we present VastGaussian, the first method for high-quality reconstruction and real-time rendering on large scenes based on 3D Gaussian Splatting. We propose a progressive partitioning strategy to divide a large scene into multiple cells, where the training cameras and point cloud are properly distributed with an airspace-aware visibility criterion. These cells are merged into a complete scene after parallel optimization. We also introduce decoupled appearance modeling into the optimization process to reduce appearance variations in the rendered images. Our approach outperforms existing NeRF-based methods and achieves state-of-the-art results on multiple large scene datasets, enabling fast optimization and high-fidelity real-time rendering. Project page: https://vastgaussian.github.io.

1. Introduction

Large scene reconstruction is essential for many applications, including autonomous driving [22, 33, 54], aerial surveying [6, 13], and virtual reality, which require photo-realistic visual quality and real-time rendering. Some approaches [41, 44, 52, 53, 61] are introduced to extend neural radiance fields (NeRF) [31] to large-scale scenes, but they still lack details or render slowly. Recently, 3D Gaussian Splatting (3DGS) [21] emerges as a promising approach with impressive performance in visual quality and rendering speed, enabling photo-realistic and real-time rendering at 1080p resolution. It is also applied to dynamic scene reconstruction [28, 51, 55, 56] and 3D content generation [12, 42, 59]. However, these methods focus on small-scale and object-centric scenes. When applied to large-scale environments, there are several scalability issues. *First*, the number of 3D Gaussians is limited by a given video memory, while the rich details of a large scene require numerous 3D Gaussians. Naively applying 3DGS to a large-scale scene would result in either low-quality reconstruction or out-of-memory errors. For intuitive explanation, a 32 GB GPU can be used to optimize about 11 million 3D Gaussians, while the small *Garden* scene in the Mip-NeRF 360

dataset [3] with an area of less than $100m^2$ already requires approximately 5.8 million 3D Gaussians for a high-fidelity reconstruction. *Second*, it requires sufficient iterations to optimize an entire large scene as a whole, which could be time-consuming, and unstable without good regularizations. *Third*, the illumination is usually uneven in a large scene, and there are noticeable appearance variations in the captured images, as shown in Fig. 2(a). 3DGS tends to produce large 3D Gaussians with low opacities to compensate for these disparities across different views. For example, bright blobs tend to come up close to the cameras with images of high exposure, and dark blobs are associated with images of low exposure. These blobs turn to be unpleasant floaters in the air when observed from novel views, as shown in Fig. 2(b, d).

To address these issues, we propose Vast 3D Gaussians (VastGaussian) for large scene reconstruction based on 3D Gaussian Splatting. We reconstruct a large scene in a divide-and-conquer manner: Partition a large scene into multiple cells, optimize each cell independently, and finally merge them into a full scene. It is easier to optimize these cells due to their finer spatial scale and smaller data size. A natural and naive partitioning strategy is to distribute training data geographically based on their positions. This may cause boundary artifacts between two adjacent cells due to few common cameras, and can produce floaters in the air without sufficient supervision. Thus, we propose visibility-based data selection to incorporate more training cameras and point clouds progressively, which ensures seamless merging and eliminates floaters in the air. Our approach allows better flexibility and scalability than 3DGS. Each of these cells contains a smaller number of 3D Gaussians, which reduces the memory requirement and optimization time, especially when optimized in parallel with multiple GPUs. The total number of 3D Gaussians contained in the merged scene can greatly exceed that of the scene trained as a whole, improving the reconstruction quality. Besides, we can expand the scene by incorporating new cells or fine-tune a specific region without retraining the entire large scene.

To reduce the floaters caused by appearance variations, Generative Latent Optimization (GLO) [5] with appearance embeddings [29] is proposed for NeRF-based methods [41, 61]. This approach samples points through ray-marching, and the point features are fed into an MLP along with appearance embeddings to obtain the final colors. The rendering process is the same as the optimization, which still requires appearance embeddings as input. It is not suitable for 3DGS as its rendering is performed by frame-wise rasterization without MLPs. Therefore, we propose a novel decoupled appearance modeling that is applied only in the optimization. We attach an appearance embedding to the rendered image pixel-by-pixel, and feed them into a CNN

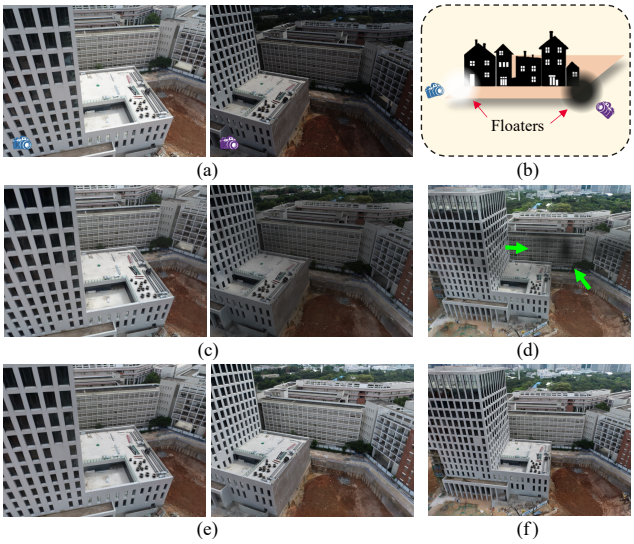

Figure 2. (a) Appearance may vary in adjacent training views. (b) Dark or bright blobs may be created near cameras with training images of different brightnesses. (c) 3D Gaussian Splatting uses these blobs to fit the appearance variations, making the renderings similar to the training images in (a). (d) These blobs appear as floaters in novel views. (e) Our decoupled appearance modeling enables the model to learn constant colors, so the rendered images are more consistent in appearance across different views. (f) Our approach greatly reduces floaters in novel views.

to obtain a transformation map for applying appearance adjustment on the rendered image. We penalize the structure dissimilarities between the rendered image and its ground truth to learn constant information, while the photometric loss is calculated on the adjusted image to fit the appearance variations in the training image. Only the consistent rendering is what we need, so this appearance modeling module can be discarded after optimization, thus not slowing down the real-time rendering speed.

Experiments on several large scene benchmarks confirm the superiority of our method over NeRF-based methods. Our contributions are summarized as follows:

- We present VastGaussian, the first method for high-fidelity reconstruction and real-time rendering on large scenes based on 3D Gaussian Splatting.
- We propose a progressive data partitioning strategy that assigns training views and point clouds to different cells, enabling parallel optimization and seamless merging.
- We introduce decoupled appearance modeling into the optimization process, which suppresses floaters due to appearance variations. This module can be discarded after optimization to obtain the real-time rendering speed.

2. Related Work

2.1. Large Scene Reconstruction

There is significant progress in image-based large scene reconstruction over the past decades. Some works [1, 16, 23, 34, 38, 39, 62] follow a structure-from-motion (SfM) pipeline to estimate camera poses and a sparse point cloud. The following works [17, 19] produce a dense point cloud or triangle mesh from the SfM output based on multi-view stereo (MVS). As NeRF [31] becomes a popular 3D representation for photo-realistic novel-view synthesis in recent years [35], many variants are proposed to improve quality [2–4, 24, 45, 47–49, 57], increase speed [8, 9, 11, 14, 20, 32, 36, 37, 40, 43, 46, 58, 60], extend to dynamic scenes [7, 15, 18, 25, 27, 50], and so on. Some methods [41, 44, 52, 53, 61] scale it to large scenes. Block-NeRF [41] divides a city into multiple blocks and distributes training views according to their positions. Mega-NeRF [44] uses grid-based division and assigns each pixel in an image to different grids through which its ray passes. Unlike these heuristics partitioning strategies, Switch-NeRF [61] introduces a mixture-of-NeRF-experts framework to learn scene decomposition. Grid-NeRF [53] does not perform scene decomposition, but rather uses an integration of NeRF-based and grid-based methods. While the rendering quality of these methods is significantly improved over traditional ones, they still lack details and render slowly. Recently, 3D Gaussian Splatting [21] introduces an expressive explicit 3D representation with high-quality and real-time rendering at 1080p resolution. However, it is non-trivial to scale it up to large scenes. Our VastGaussian is the first one to do so with novel designs for scene partitioning, optimizing, and merging.

2.2. Varying Appearance Modeling

Appearance variation is a common problem in image-based reconstruction under changing lighting or different camera setting such as auto-exposure, auto-white-balance and tone-mapping. NRW [30] trains an appearance encoder in a data-driven manner with a contrastive loss, which takes a deferred-shading deep buffer as input and produces an appearance embedding (AE). NeRF-W [29] attaches AEs to point-based features in ray-marching, and feeds them into an MLP to obtain the final colors, which becomes a standard practice in many NeRF-based methods [41, 44, 61]. Ha-NeRF [10] makes AE a global representation across different views, and learns it with a view-consistent loss. In our VastGaussian, we concatenate AEs with rendered images, feed them into a CNN to obtain transformation maps, and use the transformation maps to adjust the rendered images to fit the appearance variations.

3. Preliminaries

In this paper, we propose VastGaussian for large scene reconstruction and rendering based on 3D Gaussian Splatting (3DGS) [21]. 3DGS represents the geometry and appearance via a set of 3D Gaussians $\mathbf{G}$. Each 3D Gaussian is characterized by its position, anisotropic covariance, opacity, and spherical harmonic coefficients for view-dependent colors. During the rendering process, each 3D Gaussian is projected to the image space as a 2D Gaussian. The projected 2D Gaussians are assigned to different tiles, sorted and alpha-blended into a rendered image in a point-based volume rendering manner [63].

The dataset used to optimize a scene contains a sparse point cloud $\mathbf{P}$ and training views $\mathbf{V} = \{(\mathcal{C}_i, \mathcal{I}_i)\}$, where $\mathcal{C}_i$ is the i-th camera, and $\mathcal{I}_i$ is the corresponding image. $\mathbf{P}$ and $\{\mathcal{C}_i\}$ are estimated by Structure-from-Motion (SfM) from $\{\mathcal{I}_i\}$. $\mathbf{P}$ is used to initialize 3D Gaussians, and $\mathbf{V}$ is used for differentiable rendering and gradient-based optimization of 3D Gaussians. For camera $\mathcal{C}_i$, the rendered image $\mathcal{I}_i^r = \mathcal{R}(\mathbf{G}, \mathcal{C}_i)$ is obtained by a differentiable rasterizer $\mathcal{R}$. The properties of 3D Gaussians are optimized with respect to the loss function between $\mathcal{I}_i^r$ and $\mathcal{I}_i$ as follows:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1(\mathcal{I}_i^r, \mathcal{I}_i) + \lambda\mathcal{L}_{\text{D-SSIM}}(\mathcal{I}_i^r, \mathcal{I}_i), \quad (1)$$

where λ is a hyper-parameter, and $\mathcal{L}_{\text{D-SSIM}}$ denotes the D-SSIM loss [21]. This process is interleaved with adaptive point densification, which is triggered when the cumulative gradient of the point reaches a certain threshold.

4. Method

3DGS [21] works well on small and object-centric scenes, but it struggles when scaled up to large environments due to video memory limitation, long optimization time, and appearance variations. In this paper, we extend 3DGS to large scenes for real-time and high-quality rendering. We propose to partition a large scene into multiple cells that are merged after individual optimization. In Sec. 4.1, we introduce a progressive data partitioning strategy with airspace-aware visibility calculation. Sec. 4.2 elaborates how to optimize individual cells, presenting our decoupled appearance modeling to capture appearance variations in images. Finally, we describe how to merge these cells in Sec. 4.3.

4.1. Progressive Data Partitioning

We partition a large scene into multiple cells and assign parts of the point cloud $\mathbf{P}$ and views $\mathbf{V}$ to these cells for optimization. Each of these cells contains a smaller number of 3D Gaussians, which is more suitable for optimization with lower memory capacity, and requires less training time when optimized in parallel. The pipeline of our progressive data partitioning strategy is shown in Fig. 3.

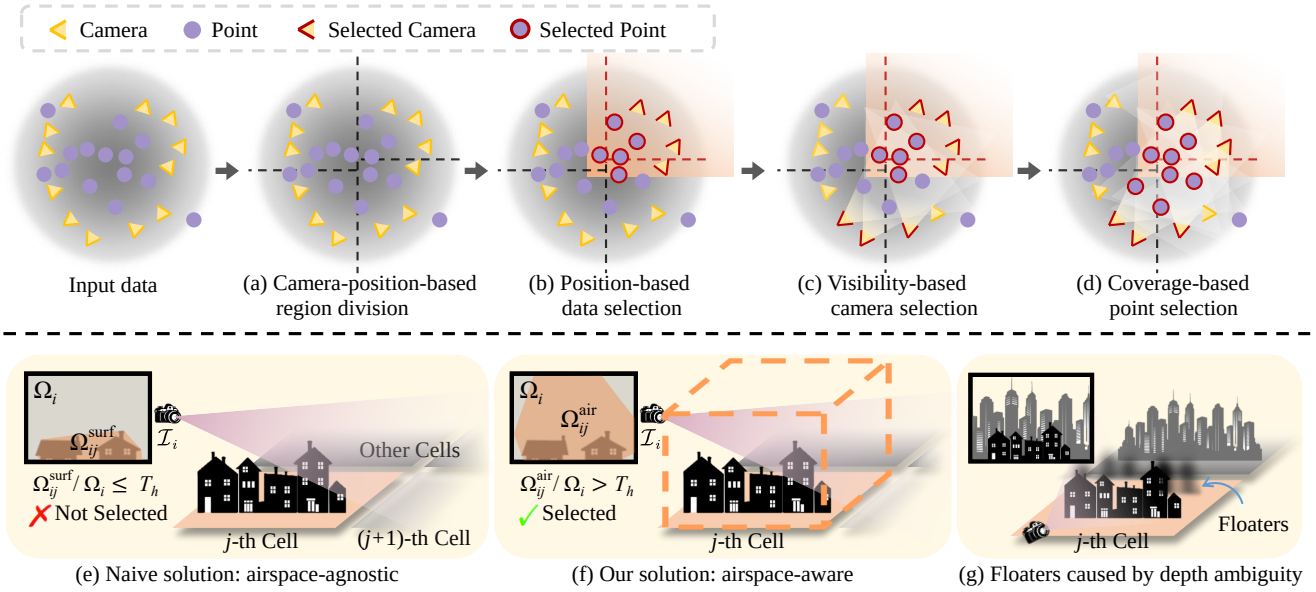

Figure 3. **Progressive data partitioning**. **Top row**: (a) The whole scene is divided into multiple regions based on the 2D camera positions projected on the ground plane. (b) Parts of the training cameras and point cloud are assigned to a specific region according to its expanded boundaries. (c) More training cameras are selected to reduce floaters, based on an airspace-aware visibility criterion, where a camera is selected if it has sufficient visibility on this region. (d) More points of the point cloud are incorporated for better initialization of 3D Gaussians, if they are observed by the selected cameras. **Bottom row**: Two visibility definitions to select more training cameras. (e) A naive way: The visibility of the i-th camera on the j-th cell is defined as $\Omega_{ij}^{\text{surf}}/\Omega_i$, where Ω_i is the area of the image $\mathcal{I}_i$, and $\Omega_{ij}^{\text{surf}}$ is the convex hull area formed by the surface points in the j-th cell that are projected to $\mathcal{I}_i$. (f) Our airspace-aware solution: The convex hull area Ω_{ij}^{air} is calculated on the projection of the j-th cell's bounding box in $\mathcal{I}_i$. (g) Floaters caused by depth ambiguity with improper point initialization, which cannot be eliminated without sufficient supervision from training cameras.

Camera-position-based region division. As shown in Fig. 3(a), we partition the scene based on the projected camera positions on the ground plane, and make each cell contain a similar number of training views to ensure balanced optimization between different cells under the same number of iterations. Without loss of generality, assuming that a grid of $m \times n$ cells fits the scene in question well, we first partition the ground plane into m sections along one axis, each containing approximately $|\mathbf{V}|/m$ views. Then each of these sections is further subdivided into n segments along the other axis, each containing approximately $|\mathbf{V}|/(m \times n)$ views. Although here we take grid-based division as an example, our data partitioning strategy is also applicable to other geography-based division methods, such as sectorization and quadtrees.

Position-based data selection. As illustrated in Fig. 3(b), we assign part of the training views $\mathbf{V}$ and point cloud $\mathbf{P}$ to each cell after expanding its boundaries. Specifically, let the j-th region be bounded in a $\ell_j^h \times \ell_j^w$ rectangle; the original boundaries are expanded by a certain percentage, 20% in this paper, resulting in a larger rectangle of size $(\ell_j^h + 0.2\ell_j^h) \times (\ell_j^w + 0.2\ell_j^w)$. We partition the training views $\mathbf{V}$ into $\{\mathbf{V}_j\}_{j=1}^{m \times n}$ based on the expanded boundaries, and segment the point cloud $\mathbf{P}$ into $\{\mathbf{P}_j\}$ in the same way.

Visibility-based camera selection. We find that the selected cameras in the previous step are insufficient for high-fidelity reconstruction, which can lead to poor detail or floater artifact. To solve this problem, we propose to add more relevant cameras based on a visibility criterion, as shown in Fig. 3(c). Given a yet-to-be-selected camera $\mathcal{C}_i$, let Ω_{ij} be the projected area of the j-th cell in the image $\mathcal{I}_i$, and let Ω_i be the area of $\mathcal{I}_i$; visibility is defined as Ω_{ij}/Ω_i. Those cameras with a visibility value greater than a predefined threshold T_h are selected.

Note that different ways of calculating Ω_{ij} result in different camera selections. As illustrated in Fig. 3(e), a natural and naive solution is based on the 3D points distributed on the object surface. They are projected on $\mathcal{I}_i$ to form a convex hull of area $\Omega_{ij}^{\text{surf}}$. This calculation is airspace-agnostic because it takes only the surface into account. Therefore some relevant cameras are not selected due to its low visibility on the j-th cell in this calculation, which results in under-supervision for airspace, and cannot suppress floaters in the air.

We introduce an airspace-aware visibility calculation, as shown in Fig. 3(f). Specifically, an axis-aligned bounding box is formed by the point cloud in the j-th cell, whose height is chosen as the distance between the highest point

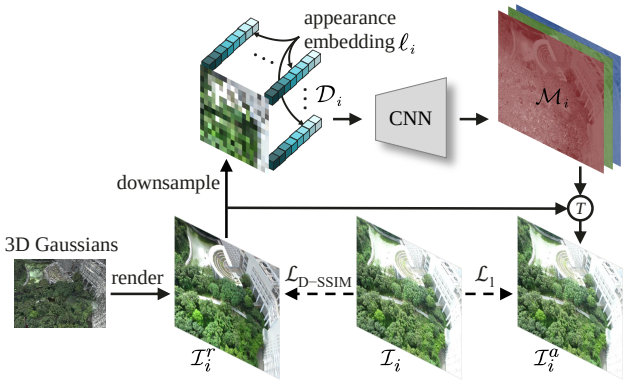

Figure 4. Decoupled appearance modeling. The rendered image $\mathcal{I}_i^r$ is downsampled to a smaller resolution, concatenated by an optimizable appearance embedding ℓ_i in a pixel-wise manner to obtain $\mathcal{D}_i$, and then fed into a CNN to generate a transformation map $\mathcal{M}_i$. $\mathcal{M}_i$ is used to perform appearance adjustment on $\mathcal{I}_i^r$ to get an appearance-variant image $\mathcal{I}_i^a$, which is used to calculate the loss $\mathcal{L}_1$ against the ground truth $\mathcal{I}_i$, while $\mathcal{I}_i^r$ is used to calculate the D-SSIM loss.

and the ground plane. We project the bounding box onto $\mathcal{I}_i$ and obtain a convex hull area Ω_{ij}^{air}. This airspace-aware solution takes into account all the visible space, which ensures that given a proper visibility threshold, the views with significant contributions to the optimization of this cell are selected and provide enough supervision for the airspace.

Coverage-based point selection. After adding more relevant cameras to the j-th cell's camera set $\mathbf{V}_j$, we add the points covered by all the views in $\mathbf{V}_j$ into $\mathbf{P}_j$, as illustrated in Fig. 3(d). The newly selected points can provide better initialization for the optimization of this cell. As illustrated in Fig. 3(g), some objects outside the j-th cell can be captured by some views in $\mathbf{V}_j$, and new 3D Gaussians are generated in wrong positions to fit these objects due to depth ambiguity without proper initialization. However, by adding these object points for initialization, new 3D Gaussians in correct positions can be easily created to fit these training views, instead of producing floaters in the j-th cell. Note that the 3D Gaussians generated outside the cell are removed after the optimization of the cell.

4.2. Decoupled Appearance Modeling

There are obvious appearance variations in the images taken in uneven illumination, and 3DGS tends to produce floaters to compensate for these variations across different views, as shown in Fig. 2(a–d).

To address this problem, some NeRF-based methods [29, 41, 44, 61] concatenate an appearance embedding to point-based features in pixel-wise ray-marching, and feed them into the radiance MLP to obtain the final color. This is not suitable for 3DGS, whose rendering is performed by frame-wise rasterization without MLPs. Instead, we introduce decoupled appearance modeling into the optimization process, which produces a transformation map to adjust the rendered image to fit the appearance variations in the training image, as shown in Fig. 4. Specifically, we first downsample the rendered image $\mathcal{I}_i^r$ to not only prevent the transformation map from learning high-frequency details, but also reduce computation burden and memory consumption. We then concatenate an appearance embedding ℓ_i of length m to every pixel in the three-channel downsampled image, and obtain a 2D map $\mathcal{D}_i$ with $3+m$ channels. $\mathcal{D}_i$ is fed into a convolutional neural network (CNN), which progressively upsamples $\mathcal{D}_i$ to generate $\mathcal{M}_i$ that is of the same resolution as $\mathcal{I}_i^r$. Finally, the appearance-variant image $\mathcal{I}_i^a$ is obtained by performing a pixel-wise transformation T on $\mathcal{I}_i^r$ with $\mathcal{M}_i$:

$$\mathcal{I}_i^a = T(\mathcal{I}_i^r; \mathcal{M}_i). \tag{2}$$

In our experiments, a simple pixel-wise multiplication works well on the datasets we use. The appearance embeddings and CNN are optimized along with the 3D Gaussians, using the loss function modified from Eq. (1):

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1(\mathcal{I}_i^a, \mathcal{I}_i) + \lambda\mathcal{L}_{\text{D-SSIM}}(\mathcal{I}_i^r, \mathcal{I}_i). \tag{3}$$

Since $\mathcal{L}_{\text{D-SSIM}}$ mainly penalizes the structural dissimilarity, applying it between $\mathcal{I}_i^r$ and the ground truth $\mathcal{I}_i$ makes the structure information in $\mathcal{I}_i^r$ close to $\mathcal{I}_i$, leaving the appearance information to be learned by ℓ_i and the CNN. The loss $\mathcal{L}_1$ is applied between the appearance-variant rendering $\mathcal{I}_i^a$ and $\mathcal{I}_i$, which is used to fit the ground truth image $\mathcal{I}_i$ that may have appearance variations from other images. After training, $\mathcal{I}_i^r$ is expected to have a consistent appearance with other images, from which the 3D Gaussians can learn an average appearance and correct geometry of all the input views. This appearance modeling can be discarded after optimization, without slowing down the real-time rendering speed.

4.3. Seamless Merging

After optimizing all the cells independently, we need to merge them to get a complete scene. For each optimized cell, we delete the 3D Gaussians that are outside the original region (Fig. 3(a)) before boundary expansion. Otherwise, they could become floaters in other cells. We then merge the 3D Gaussians of these non-overlapping cells. The merged scene is seamless in appearance and geometry without obvious border artifacts, because some training views are common between adjacent cells in our data partitioning. Therefore, there is no need to perform further appearance adjustment like Block-NeRF [41]. The total number of 3D Gaussians contained in the merged scene can greatly exceed that of the scene trained as a whole, thus improving the reconstruction quality.

Scene	*Building*			*Rubble*			*Campus*			*Residence*			*Sci-Art*		
Metrics	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS
Mega-NeRF	0.569	21.48	0.378	0.575	24.70	0.407	0.561	23.93	0.513	0.648	22.86	0.330	0.769	26.25	0.263
Switch-NeRF	0.594	22.07	0.332	0.586	24.93	0.377	0.565	24.03	0.495	0.675	23.41	0.280	0.793	**27.07**	0.224
Grid-NeRF (grid branch)	–	–	–	0.780	25.47	0.213	0.767	25.51	0.174	0.807	**24.37**	0.142	–	–	–
Grid-NeRF (nerf branch)	–	–	–	0.767	24.13	0.207	0.757	24.90	0.162	0.802	23.77	0.137	–	–	–
Modified 3DGS	0.769	23.01	0.164	0.800	26.78	0.161	0.712	23.89	0.289	0.825	23.40	0.142	0.843	25.24	0.166
VastGaussian (Ours)	**0.804**	**23.50**	**0.130**	**0.823**	**26.92**	**0.132**	**0.816**	**26.00**	**0.151**	**0.852**	24.25	**0.124**	**0.885**	26.81	**0.121**

Table 1. Quantitative evaluation of our method compared to previous work on five large scenes. We report SSIM↑, PSNR↑ and LPIPS↓ on test views. The **best** and second best results are highlighted. "–" denotes missing data from the Grid-NeRF paper.

5. Experiments

5.1. Experimental Setup

Implementation. We evaluate our model with 8 cells in our main experiments. The visibility threshold is 25%. The rendered images are downsampled by 32 times before being concatenated with the appearance embeddings of length 64. Each cell is optimized for 60,000 iterations. The densification [21] starts at the 1,000th iteration and ends at the 30,000th iteration, with an interval of 200 iterations. The other settings are identical to those of 3DGS [21]. Both the appearance embeddings and the CNN use a learning rate of 0.001. We perform Manhattan world alignment to make the y-axis of the world coordinate perpendicular to the ground plane. We describe the CNN architecture in the supplement.

Datasets. The experiments are conducted on five large-scale scenes: *Rubble* and *Building* from the Mill-19 dataset [44], and *Campus*, *Residence*, and *Sci-Art* from the UrbanScene3D dataset [26]. Each scene contains thousands of high-resolution images. We downsample the images by 4 times for training and validation, following previous methods [44, 61] for fair comparison.

Metrics. We evaluate the rendering quality using three quantitative metrics: SSIM, PSNR, and AlexNet-based LPIPS. The aforementioned photometric variation makes the evaluation difficult, as it is uncertain which photometric condition should be replicated. To address this issue, we follow Mip-NeRF 360 [3] to perform color correction on the rendered images before evaluating the metrics of all the methods, which solves a per-image least squares problem to align the RGB values between the rendered image and its corresponding ground truth. We also report the rendering speed at 1080p resolution, average training time, and video memory consumption.

Compared methods. We compare our VastGaussian with four methods: Mega-NeRF [44], Switch-NeRF [61], Grid-NeRF [53], and 3DGS [21]. For 3DGS, we need to increase optimization iterations to make it comparable in our main experiments, but naively doing that causes out-of-memory errors. Therefore, we increase the densification interval accordingly to build a feasible baseline (termed Modified 3DGS). The other configurations are the same as those in the original 3DGS paper. For Grid-NeRF, its code is released without rendered images and carefully tuned configuration files due to its confidentiality requirements. These unavailable files are critical to its performance, making its results not reproducible. Therefore, we use its code to evaluate only its training time, memory and rendering speed, while the quality metrics are copied from its paper.

Dataset	Mill-19			UrbanScene3D		
Metrics	Training	VRAM	FPS	Training	VRAM	FPS
Mega-NeRF	30h19m	**5.0G**	0.25	28h01m	**5.0G**	0.31
Switch-NeRF	41h58m	9.9G	0.12	38h47m	9.9G	0.15
Grid-NeRF	17h44m	14.0G	0.26	17h38m	14.0G	0.30
Modified 3DGS	19h32m	31.1G	**177.75**	20h12m	31.2G	**210.99**
VastGaussian	**2h25m**	10.4G	126.45	**2h56m**	11.9G	171.92

Table 2. Comparison of training time, training video memory consumption (VRAM), and rendering speed.

5.2. Result Analysis

Reconstruction quality. In Tab. 1, we report the mean SSIM, PSNR, and LPIPS metrics in each scene. Our VastGaussian outperforms the compared methods in all the SSIM and LPIPS metrics by significant margins, suggesting that it reconstructs richer details with better rendering in perception. In terms of PSNR, VastGaussian achieves better or comparable results. We also show visual comparison in Fig. 5. The NeRF-based methods fall short of details and produce blurry results. Modified 3DGS has sharper rendering but produces unpleasant floaters. Our method achieves clean and visually pleasing rendering. Note that due to the noticeable over-exposure or under-exposure in some test images, VastGaussian exhibits slightly lower PSNR values, but produces significantly better visual quality, sometimes even being more clear than the ground truth, such as the example on the 3rd row in Fig. 5. The high quality of VastGaussian is partly thanks to its large number of 3D Gaussians. Take the *Campus* scene for example, the number of 3D Gaussians in Modified 3DGS is 8.9 million, while for VastGaussian the number is 27.4 million.

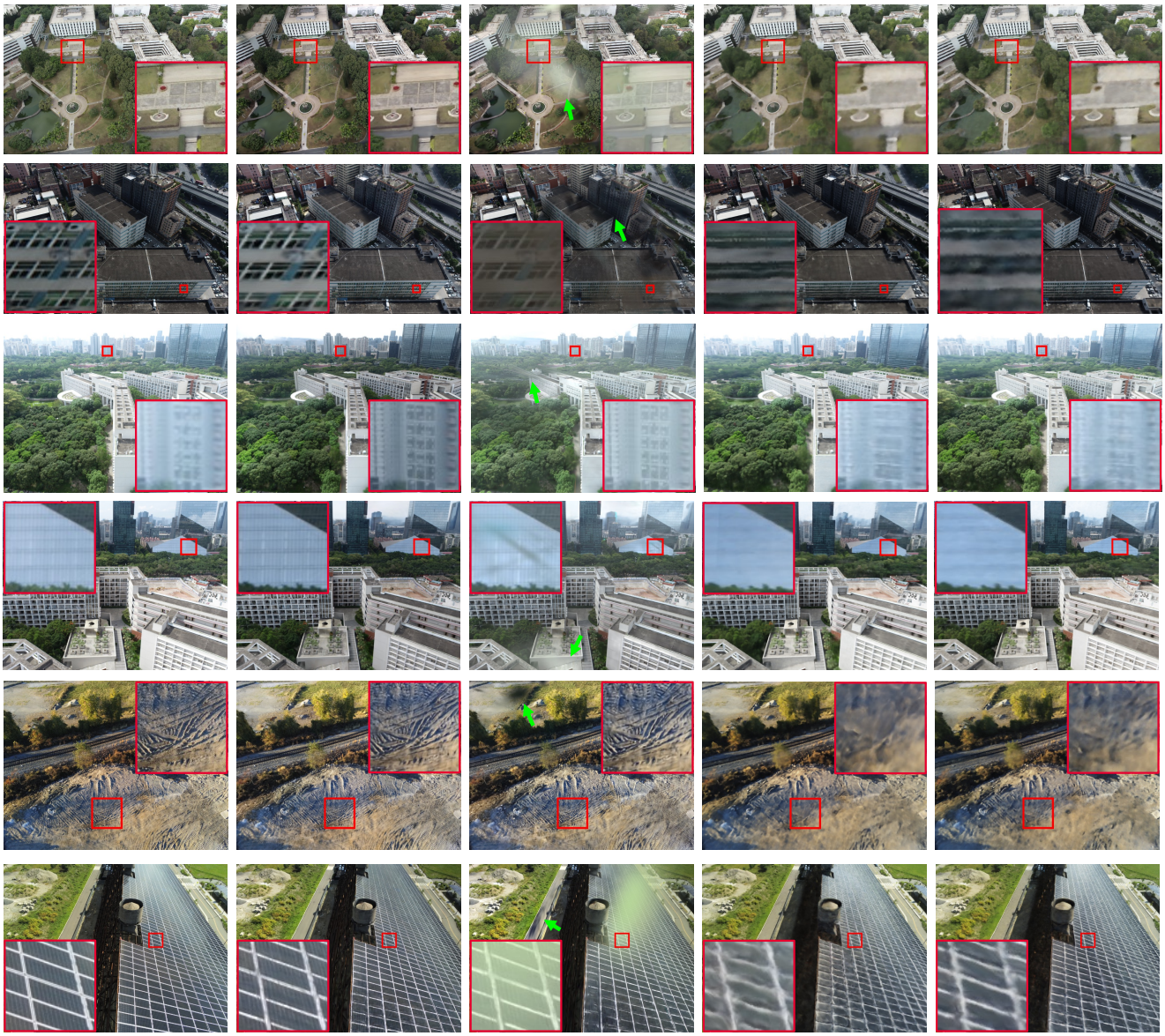

Figure 5. Qualitative comparison between VastGaussian and previous work. Floaters are pointed out by green arrows.

Efficiency and memory. In Tab. 2, we report the training time, video memory consumption during optimization, and rendering speed. Mega-NeRF, Switch-NeRF and VastGaussian are trained on 8 Tesla V100 GPUs, while Grid-NeRF and Modified 3DGS on a single V100 GPU as they do not perform scene decomposition. The rendering speeds are tested on a single RTX 3090 GPU. Our VastGaussian requires much shorter time to reconstruct a scene with photo-realistic rendering. Compared to Modified 3DGS, VastGaussian greatly reduces video memory consumption on a single GPU. Since VastGaussian has more 3D Gaussians in the merged scene than Modified 3DGS, its rendering speed is slightly slower than Modified 3DGS, but is still much faster than the NeRF-based methods, achieving real-time rendering at 1080p resolution.

5.3. Ablation Study

We perform ablation study on the *Sci-Art* scene to evaluate different aspects of VastGaussian.

Data partition. As shown in Fig. 6 and Tab. 3, both visibility-based camera selection (VisCam) and coverage-based point selection (CovPoint) can improve visual quality. Without each or both of them, floaters can be created in the airspace of a cell to fit the views observing regions out-

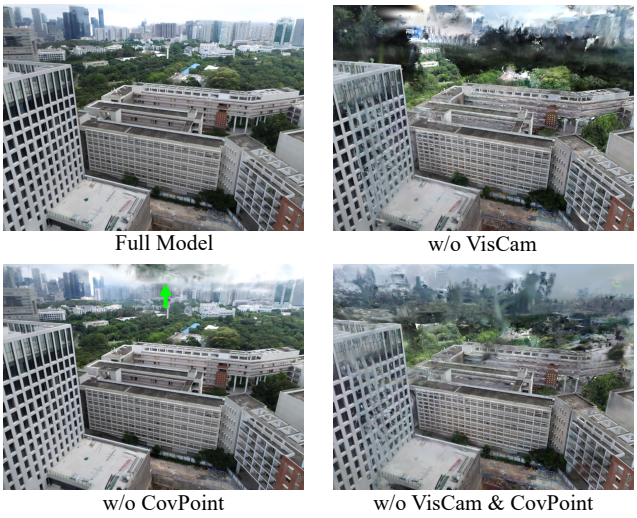

Figure 6. The visibility-based camera selection and coverage-based point selection can reduce floaters in the airspace.

Model setting	SSIM	PSNR	LPIPS
1) w/o VisCam	0.694	20.05	0.261
2) w/o CovPoint	0.874	26.14	0.128
3) w/o VisCam & CovPoint	0.699	20.35	0.253
4) airspace-aware → agnostic	0.855	24.54	0.128
5) w/o Decoupled AM	0.858	25.08	0.148
Full model	**0.885**	**26.81**	**0.121**

Table 3. Ablation on data partition, visibility calculation and decoupled appearance modeling (Decoupled AM).

side the cell. As shown in Fig. 7, the visibility-based camera selection can ensure more common cameras between adjacent cells, which eliminates the noticeable boundary artifact of appearance jumping when it is not implemented.

Airspace-aware visibility calculation. As illustrated in the 4th row of Tab. 3 and Fig. 8, the cameras selected based on the airspace-aware visibility calculation provide more supervision for the optimization of a cell, thus not producing floaters that are presented when the visibility is calculated in the airspace-agnostic way.

Decoupled appearance modeling. As shown in Fig. 2 and the 5th row of Tab. 3, our decoupled appearance modeling reduces the appearance variations in the rendered images. Therefore, the 3D Gaussians can learn consistent geometry and colors from training images with appearance variations, instead of creating floaters to compensate for these variations. Please also see the videos in the supplement.

Different number of cells. As shown in Tab. 4, more cells reconstruct better details in VastGaussian, leading to better SSIM and LPIPS values, and shorter training time when the cells are optimized in parallel. However, when the cell number reaches 16 or bigger, the quality improvement becomes

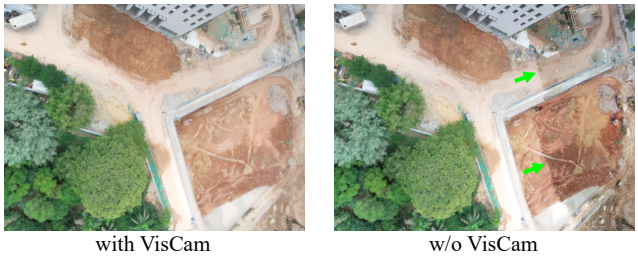

Figure 7. The visibility-based camera selection can eliminate the appearance jumping on the cell boundaries.

Figure 8. Heavy floaters appear when the visibility is calculated in the airspace-agnostic way.

#Cell	#GPU	SSIM	PSNR	LPIPS	Training
4	4	0.870	26.39	0.136	2h46m
8	8	0.885	**26.81**	0.121	2h39m
16	16	0.888	26.80	0.116	2h30m
24	24	**0.892**	26.64	**0.110**	**2h19m**

Table 4. Effect of different cell numbers.

marginal, and PSNR slightly decreases because there may be gradual brightness changes in a rendered image from cells that are far apart.

6. Conclusion and Limitation

In this paper, we propose VastGaussian, the first high-quality reconstruction and real-time rendering method on large-scale scenes. The introduced progressive data partitioning strategy allows for independent cell optimization and seamless merging, obtaining a complete scene with sufficient 3D Gaussians. Our decoupled appearance modeling decouples appearance variations in the training images, and enables consistent rendering across different views. This module can be discarded after optimization to obtain faster rendering speed. While our VastGaussian can be applied to spatial divisions of any shape, we do not provide an optimal division solution that should consider the scene layout, the cell number and the training camera distribution. In addition, there are a lot of 3D Gaussians when the scene is huge, which may need a large storage space and significantly slow down the rendering speed.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 2011. 3

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021. 3

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 2, 6

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

[5] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. In *ICML*, 2018. 2

[6] Ilker Bozcan and Erdal Kayacan. Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance. In *ICRA*, 2020. 1

[7] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *CVPR*, 2023. 3

[8] Junli Cao, Huan Wang, Pavlo Chemerys, Vladislav Shakhrai, Ju Hu, Yun Fu, Denys Makoviichuk, Sergey Tulyakov, and Jian Ren. Real-time neural light field on mobile devices. In *CVPR*, 2023. 3

[9] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 3

[10] Xingyu Chen, Qi Zhang, Xiaoyu Li, Yue Chen, Ying Feng, Xuan Wang, and Jue Wang. Hallucinated neural radiance fields in the wild. In *CVPR*, 2022. 3

[11] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *CVPR*, 2023. 3

[12] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585*, 2023. 1

[13] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. The unmanned aerial vehicle benchmark: Object detection and tracking. In *ECCV*, 2018. 1

[14] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 3

[15] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 3

[16] Christian Früh and Avideh Zakhor. An automated method for large-scale, ground-based city model acquisition. *IJCV*, 2004. 3

[17] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010. 3

[18] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, 2022. 3

[19] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *ICCV*, 2007. 3

[20] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *ICCV*, 2021. 3

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM ToG*, 2023. 1, 3, 6

[22] Wei Li, CW Pan, Rong Zhang, JP Ren, YX Ma, Jin Fang, FL Yan, QC Geng, XY Huang, HJ Gong, et al. Aads: Augmented autonomous driving simulation using data-driven algorithms. *Science robotics*, 2019. 1

[23] Xiaowei Li, Changchang Wu, Christopher Zach, Svetlana Lazebnik, and Jan-Michael Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *ECCV*, 2008. 3

[24] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023. 3

[25] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia*, 2022. 3

[26] Liqiang Lin, Yilin Liu, Yue Hu, Xingguang Yan, Ke Xie, and Hui Huang. Capturing, reconstructing, and simulating: the urbanscene3d dataset. In *ECCV*, 2022. 1, 6

[27] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *CVPR*, 2023. 3

[28] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 1

[29] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 2, 3, 5

[30] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *CVPR*, 2019. 3

[31] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 3

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM ToG*, 2022. 3

[33] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *CVPR*, 2021. 1

[34] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed real-time urban 3d reconstruction from video. *IJCV*, 2008. 3

[35] Ravi Ramamoorthi. Nerfs: The search for the best 3d representation. *arXiv preprint arXiv:2308.02751*, 2023. 3

[36] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 3

[37] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM ToG*, 2023. 3

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

[39] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH*. 2006. 3

[40] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 3

[41] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *CVPR*, 2022. 1, 2, 3, 5

[42] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 1

[43] Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. Delicate textured mesh recovery from nerf via adaptive surface refinement. In *ICCV*, 2023. 3

[44] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *CVPR*, 2022. 1, 3, 5, 6

[45] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *CVPR*, 2022. 3

[46] Huan Wang, Jian Ren, Zeng Huang, Kyle Olszewski, Menglei Chai, Yun Fu, and Sergey Tulyakov. R2l: Distilling neural radiance field to neural light field for efficient novel view synthesis. In *ECCV*, 2022. 3

[47] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 3

[48] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *CVPR*, 2023.

[49] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *ICCV*, 2023. 3

[50] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *CVPR*, 2022. 3

[51] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 1

[52] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *ECCV*, 2022. 1, 3

[53] Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. In *CVPR*, 2023. 1, 3, 6

[54] Zhenpei Yang, Yuning Chai, Dragomir Anguelov, Yin Zhou, Pei Sun, Dumitru Erhan, Sean Rafferty, and Henrik Kretzschmar. Surfelgan: Synthesizing realistic sensor data for autonomous driving. In *CVPR*, 2020. 1

[55] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 1

[56] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 1

[57] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021. 3

[58] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv preprint arXiv:2302.14859*, 2023. 3

[59] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*, 2023. 1

[60] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 3

[61] MI Zhenxing and Dan Xu. Switch-nerf: Learning scene decomposition with mixture of experts for large-scale neural radiance fields. In *ICLR*, 2022. 1, 2, 3, 5, 6

[62] Siyu Zhu, Runze Zhang, Lei Zhou, Tianwei Shen, Tian Fang, Ping Tan, and Long Quan. Very large-scale global sfm by distributed motion averaging. In *CVPR*, 2018. 3

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

VastGaussian: Vast 3D Gaussians for Large Scene Reconstruction
Supplementary Material

We elaborate more details of decoupled appearance modeling, including the CNN architecture, training time and memory, exploration of more complex transformations, and visualization of the transformation maps.

7. Details of Decoupled Appearance Modeling

7.1. CNN Architecture

The CNN architecture in the decoupled appearance modeling is shown in Fig. 9. We downsample the rendered image of shape $H \times W \times 3$ by 32 times, and then concatenate an appearance embedding of length 64 to every pixel in the downsampled image to obtain a feature map $\mathcal{D}_i$ of shape $\frac{H}{32} \times \frac{W}{32} \times 67$, which is fed to the CNN.

$\mathcal{D}_i$ is first passed through a 3×3 convolutional layer to increase its channel depth to 256. Then it is processed by 4 upsampling blocks, each containing an upsampling pixel shuffle layer, a 3×3 convolutional layer, and ReLU activation. Each upsampling block doubles the feature map resolution and halves its channel depth. After that, a bilinear interpolation layer upsamples the feature map to $H \times W$, and two 3×3 convolutional layers with a ReLU in between are followed to obtain a transformation map of shape $H \times W \times 3$, which is later used to perform appearance adjustment on the rendered image.

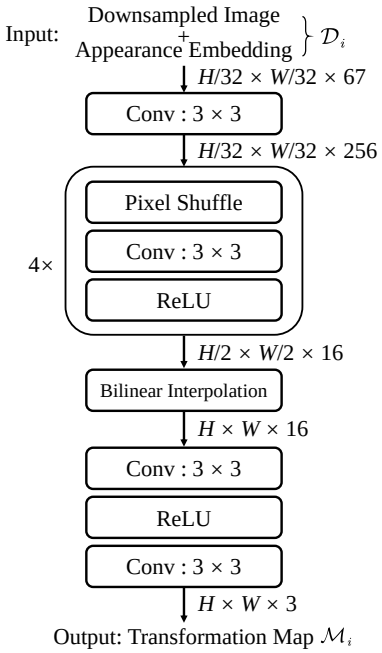

Figure 9. CNN architecture of decoupled appearance modeling.

7.2. Training Time and Memory

We provide the training time and video memory consumption to complement the ablation study of the decoupled appearance modeling in Tab. 5. Our model with the decoupled appearance modeling takes less than 10% more time and video memory to optimize due to the introduction of the appearance embeddings and the CNN, but it significantly improves rendering quality.

Model setting	SSIM	PSNR	LPIPS	Training	VRAM
w/o Decoupled AM	0.858	25.08	0.148	**1h25m**	**10.23G**
w/ Decoupled AM	**0.885**	**26.81**	**0.121**	1h33m	11.18G

Table 5. Quality metrics, training time, and video memory in the ablation study of decoupled appearance modeling on the *Sci-Art* scene.

7.3. More Complex Transformations

A simple pixel-wise multiplication works well on the datasets we use. As shown in Tab. 6, we can also extend it to the affine transformation or add some prior knowledge such as Gamma correction. The CNN architecture is accordingly adjusted to output a 6-channel map (for the multiplication plus addition) or a 4-channel map (with one additional channel for Gamma correction). Both extensions have marginal effect on the results, which indicate that the simple pixel-wise multiplication is sufficient for the datasets used in the experiments.

Transformation	SSIM	PSNR	LPIPS
multiplication	0.885	26.81	0.116
multiplication + addition	0.886	26.80	0.116
multiplication + Gamma correction	0.885	26.83	0.115

Table 6. More complex transformations of decoupled appearance modeling on the *Sci-Art* scene.

7.4. Transformation Visualization

In the main paper, we show that our decoupled appearance modeling enables VastGaussian to learn constant colors between adjacent training views. For two adjacent training images $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$, here we visualize their transformation maps $\mathcal{M}_i$ and $\mathcal{M}_{i+1}$, rendered images $\mathcal{I}_i^r$ and $\mathcal{I}_{i+1}^r$, and adjusted images $\mathcal{I}_i^a$ and $\mathcal{I}_{i+1}^a$, as shown in Fig. 10. As the optimization proceeds, the appearance variations between $\mathcal{I}_i^r$ and $\mathcal{I}_{i+1}^r$ are reduced, the appearance transformation maps $\mathcal{M}_i$ and $\mathcal{M}_{i+1}$ gradually converge, and the adjusted images $\mathcal{I}_i^a$ and $\mathcal{I}_{i+1}^a$ finally fit the training images $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$. Note that for simplicity, we turn the 3-channel transformation maps into grayscale images in the visualization.

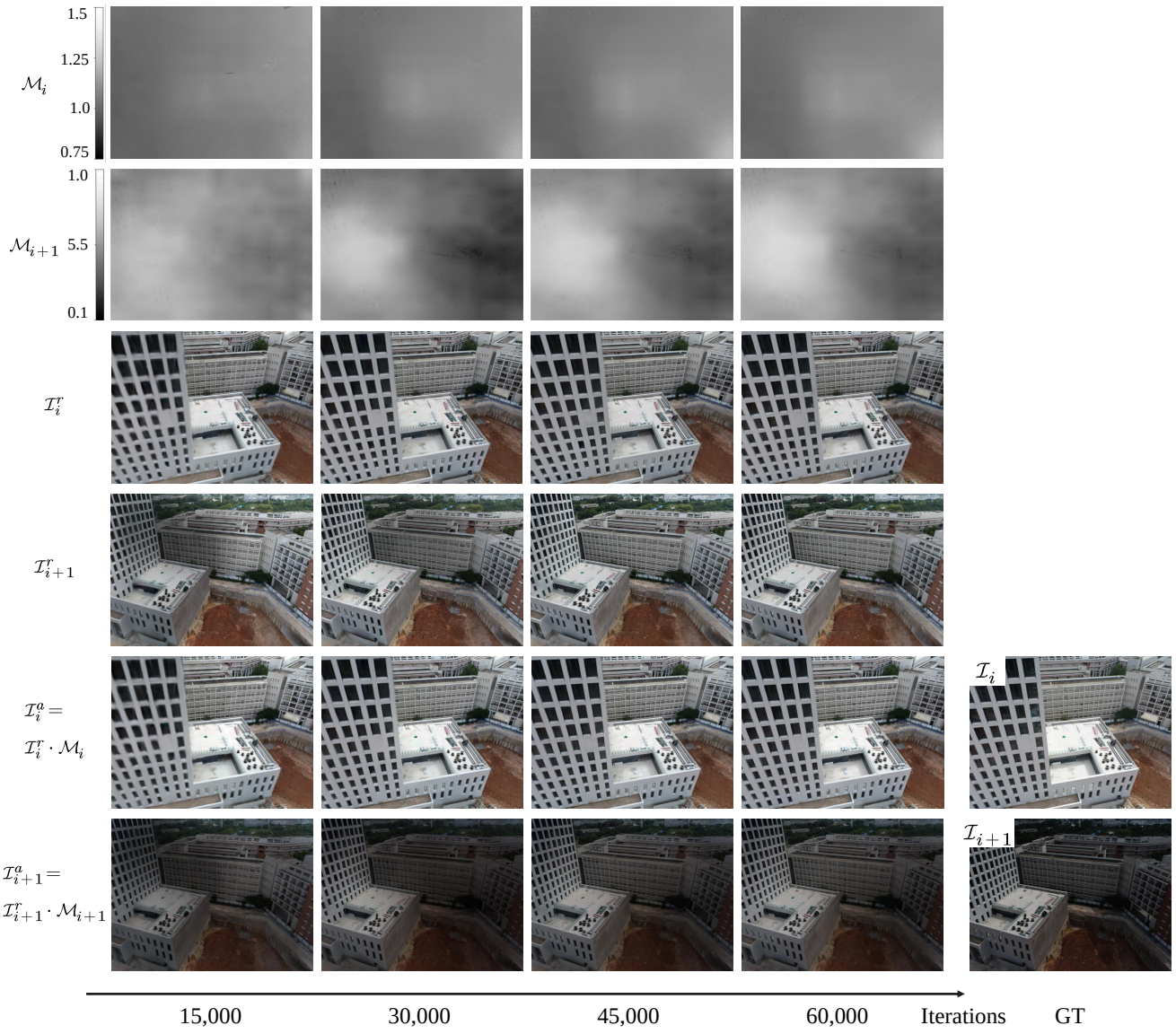

Figure 10. Evolution of the transformation maps ($\mathcal{M}_i$ and $\mathcal{M}_{i+1}$), the rendered images ($\mathcal{I}_i^r$ and $\mathcal{I}_{i+1}^r$), and the adjusted images ($\mathcal{I}_i^a$ and $\mathcal{I}_{i+1}^a$) during the optimization process.