ULSR-GS: Ultra Large-scale Surface Reconstruction Gaussian Splatting with Multi-View Geometric Consistency

Zhuoxiao Li [1,2,*]   Shanliang Yao [1,2,*]   Qizhong Gao [1,2]   Ángel F. García-Fernández [1,3]
Yong Yue [1,2]   Xiaohui Zhu [1,2]

[1] University of Liverpool, [2] Xi'an Jiaotong-Liverpool University,
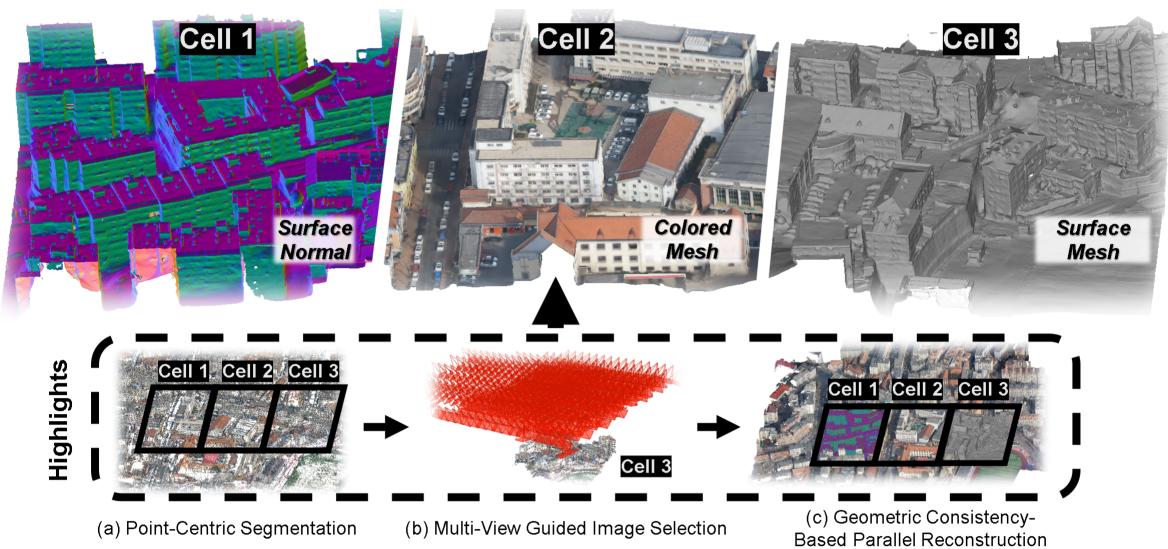[3] ARIES Research Centre, Universidad Antonio de Nebrija

Figure 1. We present **ULSR-GS**, a large-scale high-fidelity surface extraction framework based on Gaussian Splatting. The three sub-regions shown in the figure are stitched together after parallel training by three GPUs on the *Hospital* scene of the UrbanScene3D [21] dataset.

Abstract

*While Gaussian Splatting (GS) demonstrates efficient and high-quality scene rendering and small area surface extraction ability, it falls short in handling large-scale aerial image surface extraction tasks. To overcome this, we present **ULSR-GS**, a framework dedicated to high-fidelity surface extraction in ultra-large-scale scenes, addressing the limitations of existing GS-based mesh extraction methods. Specifically, we propose a point-to-photo partitioning approach combined with a multi-view optimal view matching principle to select the best training images for each sub-region. Additionally, during training, ULSR-GS employs a densification strategy based on multi-view geometric consistency to enhance surface extraction details. Experimental results demonstrate that ULSR-GS outperforms other state-of-the-art GS-based works on large-scale aerial photogrammetry benchmark datasets, significantly improving surface extraction accuracy in complex urban environments. Project page: https://ulsrgs.github.io.*

1. Introduction

3D Gaussian Splatting (3DGS) has emerged as a ground-breaking approach in 3D surface reconstruction and rendering, enabling high-quality scene representations and supporting the extraction of detail-rich meshes [12]. However, existing GS-based research has primarily focused on small-scale surface reconstructions [3, 10, 11, 18, 49] or large-scale scene rendering [19, 22, 48] without optimizing for large-scale urban surface reconstruction. Several factors contribute to the scarcity of GS-based applications in large-scale urban surface reconstruction: (1) Computational complexity is a significant hurdle [19, 24], as processing vast urban datasets on a single GPU demands extensive computational resources and memory, challenging the scalability of

1

GS methods. (2) Existing large-scale GS-based methods are typically optimized for scene rendering, and their partition strategies lack the necessary adaptations for handling the mesh reconstruction tasks. (3) Reconstruction quality challenges arise due to insufficient densification processes in certain areas, especially when handling thousands of aerial oblique images [7, 8].

To tackle the challenges of GS-based large-scale surface reconstruction, we propose ULSR-GS, a large-scale surface reconstruction method combined with the point-to-photo partition method and multi-view constrained densification. Specifically, unlike methods that focus on image position-based partitioning [19, 48], we select the optimal set of views based on the matched multi-view images [20, 36, 39, 41, 42, 45], considering factors such as camera angle, proximity, and pairing quality for each point. This targeted image selection not only ensures high-quality surface reconstruction by leveraging the most informative images but also reduces redundant data processing. We also introduce a multi-view training strategy that significantly enhances the reconstruction quality. During each training iteration, we impose additional constraints based on the best-matched views corresponding to the current image being processed [18, 42]. This training strategy enables more robust and consistent reconstructions by leveraging the most informative images from multiple angles. We further incorporate this multi-view consistency constraint in the densification process. This ensures that the densification step adheres to consistent geometric relationships across views, leading to more accurate surface details.

Our contributions are summarized as follows:

- We present ULSR-GS, a novel method specifically designed to overcome the limitations of existing GS-based works in large-scale surface reconstruction.
- We propose an innovative point-to-photo scene partition strategy for GS in large-scale scene mesh extraction. This method select images for each sub-region based on the best-matched views, improving reconstruction accuracy.
- We propose multi-view consistency constraints for the additional densification process for generating detailed and accurate reconstructions in large-scale urban environments.

2. Related work

2.1. Large-Scale Novel View Synthesis

Recent developments in novel view synthesis [1, 5, 26, 50] include several innovative approaches tailored for large-scale environments [25, 31, 32, 37]. BungeeNeRF first addresses the challenge of rendering city-scale scenes with drastic scale variations by introducing a progressive neural radiance field that adapts to different levels of detail [37]. BlockNeRF segments a city into distinct blocks and strate-

gically allocates training views based on their spatial locations [31]. Mega-NeRF takes a grid-based approach, assigning each pixel in an image to various grids corresponding to the rays it traverses [32]. In contrast to these partitioning methods, Switch-NeRF employs a mixture-of-experts framework that facilitates scene decomposition, allowing for more nuanced representation [25]. Grid-NeRF, while not focusing on scene decomposition, combines NeRF-based techniques with grid-based methodologies to enhance rendering efficiency [40].

Moreover, the adoption of 3D Gaussian Splatting [12] has emerged as a promising approach, improving both reconstruction accuracy and rendering speed [8, 24, 46]. GS-based large-scale rendering methods primarily handle large-scale scene rendering through camera-position-based partitioning, followed by point cloud selection or segmentation [4, 13, 19, 22, 48]. This strategy ensures that training images for each region remain as similar as possible, enabling efficient techniques like Level of Detail (LOD) to accelerate rendering [4, 13, 22]. This partitioning approach is particularly effective for terrestrial sequences [13] and aerial five-directional oblique photography [4, 19, 48]. However, when dealing with scenarios involving specific path planning algorithms, these partitioning methods may struggle to adapt efficiently, leading to suboptimal training data distribution and reduced reconstruction quality for dynamically changing viewpoints.

2.2. 3D Surface Reconstruction

3D reconstruction has evolved from traditional Multi-View Stereo (MVS) [2, 29, 41] to advanced neural methods [27, 34, 44], and mesh extraction methods transform implicit representations into explicit 3D models. Poisson Surface-based methods extract surfaces from point clouds but often produce overly smooth results, lacking fine details [10]. Marching Cubes converts volumetric data into polygonal meshes, but its grid-based nature can lead to artifacts and increased memory usage [18, 23]. Neural implicit representations encode 3D geometry continuously, enabling smooth modeling and handling complex topologies [34, 43, 44]. However, these methods require significant computational resources and often struggle with fine geometric details. Mesh extraction from these representations also suffers from discretization artifacts. Neural Radiance Fields (NeRF) use volumetric rendering to achieve high-quality view synthesis [26]. Despite their impressive results, NeRF-based methods are computationally intensive, making them less practical for real-time applications [1, 32]. Recent works extract explicit geometry from NeRF, but converting density fields to surfaces in large-scale scenes remains challenging and may produce noisy results [17].

Some research has begun to explore mesh extraction us-

ing 3DGS [12]. Early work combined 3DGS with Poisson Surface Reconstruction to achieve reconstruction [10]. Subsequently, studies used rendered depth maps with Truncated Signed Distance Function (TSDF) fusion to reconstruct surfaces [35], [33], improving accuracy by modifying the depth calculation method [3, 11, 18, 49], and optimizing the densification process during training [9, 18]. Additionally, a study utilized Gaussian opacity fields combined with a marching tetrahedra approach to achieve high-quality reconstruction results [47].Existing GS-based methods perform inadequately in large-scale scenes, struggling to accurately reconstruct intricate geometric details and prone to generating significant artifacts. Using the partition rendering methods [16, 19, 22] to split the scene into different sub-regions can solve this problem well, and each sub-region can obtain sufficient densification. However, the current partitioning methods lack optimization of the mesh extraction task and usually require mesh extraction after merging the entire scene.

3. Preliminaries

3.1. Multi-View Pair Selection

The *multi-view pair selection* utilizes a reference image along with three source images ($N = 3$) [6, 30, 36, 42, 45]. For each image pair, they calculate a score $s(i, j)$ based on sparse points. The score is defined as:

$$s(i, j) = \sum_p G(\theta_{ij}(p)) \tag{1}$$

where p represents a common track point in both views i and j, and $\theta_{ij}(p)$ is the baseline angle between the views.

here, $\mathbf{c}_i$ and $\mathbf{c}_j$ are the camera centers of views i and j, respectively. The function G is a piecewise Gaussian function that favors a certain baseline angle θ_0:

$$G(\theta) = \begin{cases} \exp\left(-\frac{(\theta - \theta_0)^2}{2\sigma_1^2}\right), \theta \le \theta_0 \\ \exp\left(-\frac{(\theta - \theta_0)^2}{2\sigma_2^2}\right), \theta > \theta_0 \end{cases} \tag{2}$$

3.2. Depth Calculation in 2D Gaussian Splatting

In 2DGS [11], two methodologies are employed for aggregating depth information.

Weighted aggregation. For each Gaussian G_i intersecting along a viewing ray, the depth D_{mean} is computed as a weighted sum:

$$D_{mean} = \sum_i \omega_i D_i / (\sum_i \omega_i D_i + \epsilon) \tag{3}$$

where ω_i represents the weight contribution of the i-th Gaussian. The result is normalized by the cumulative alpha values to obtain the expected depth.

Maximum visibility. This method focuses on selecting the depth of the most "visible" Gaussian along the viewing ray. The depth D_{median} is determined by the first Gaussian that contributes significantly to the cumulative alpha:

$$D_{median} = \max\{z_i | T_i > 0.5\} \tag{4}$$

where T_i quantifies the visibility based on the i-th Gaussian's alpha value.

D_{mean} facilitates smoother depth calculations, whereas D_{median} more effectively captures rich geometric details. however, it is highly susceptible to floating-point errors that can impact depth computation. In this paper, we adopt D_{mean} as the depth representation for subsequent analyses.

4. Method

We introduce the proposed point partition method with multi-view image selection in Section 4.1. Section 4.2 introduces a multi-view densification method with depth aggregation. In Section 4.3, we introduce the loss function for the multi-view training process.

4.1. Multi-View Optimized Point Partitioning

Different from previous studies that use drone photo locations for region partitioning [19, 22, 48], our scene partitioning method is based on the initial point cloud of the scene and selects the best training images for the point cloud in each sub-region. The advantage of this method is that we can determine the boundaries of the mesh to be extracted for each sub-area at the early stage, without the need to merge the full scene.

Density-controlled boundary refinement. To effectively partition the initial points into $n \times n$ sub-regions, it is crucial to isolate the primary structural components of the scene and eliminate sparse and noisy SfM points that could distort the boundary definition.
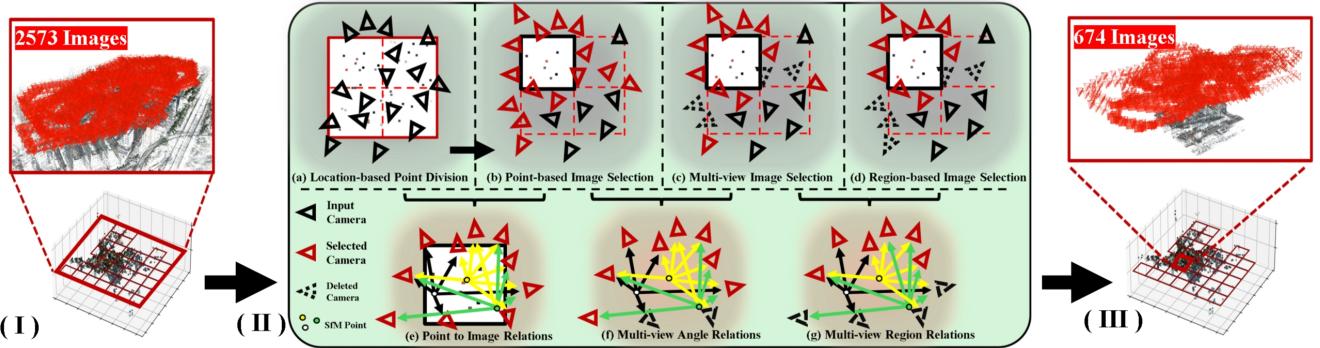
We first remove all 3D points with a SfM reprojection error [29] greater than a threshold ϵ_{error} ($\epsilon_{error} > 1.5$ in our experiments). This step cleans up the point cloud by discarding unreliable points that could skew the partitioning. Then we partition the 3D space into a voxel grid v_i with size ($s = 2$), and each SfM point $p_i = (x_i, y_i, z_i)$ is assigned to a voxel based on its coordinates:

$$v_i = \left(\left\lfloor \frac{x_i}{s} \right\rfloor, \left\lfloor \frac{y_i}{s} \right\rfloor, \left\lfloor \frac{z_i}{s} \right\rfloor\right). \tag{5}$$

For each voxel v, we count the number of points N_v it contains:

$$N_v = \sum_{i=1}^{N} \delta(v_i, v), \tag{6}$$

where N is the total number of points and $\delta(v_i, v)$ is the Kronecker delta function [15]. Then we compute a density

Figure 2. **Point-based partition strategy of ULSR-GS. (I)** We partition the training scene based on the positions of the point cloud projected onto the xz-plane. **(II)** For each sub-scene: *(a)* we first select all matching images within the region; *(b)* based on the camera relationships of the sub-scene, we select the corresponding images for each point; *(c)* following the principle of multi-view optimal matching [42], we filter out images with poor view angles; and finally, *(d)* considering the distance relationship, we further remove images that are too distant. **(III)** The final reconstructed sub-regions significantly reduce the number of training images.

threshold N_τ ($\tau = 33.3\%$) of the maximum voxel occupancy to identify the densely populated voxels:

$$N_\tau = \mathbf{max}_{v \in V} N_v. \tag{7}$$

Voxels with occupancy $N_v > N_\tau$ are considered dense and are retained. Finally, we can get the precise boundary of the scene by computing the minimum and maximum coordinates of the points.

Initial view selection. As shown in Figure 2 (I), the input COLMAP SfM [29] point cloud is divided into $n \times n$ grids after density filtering. Each point in the sub-region is the feature of the images in which it was detected and matched. We first select all matched images as a coarse view selection (see Figure 2 (e)).

Source view selection. To further refine the view selection in Eq. (1), a region constraint is applied based on the camera pair distance d_{ij}. Only image pairs with a distance below a specified maximum threshold $D_{\max}$ are considered for matching. The final matching score S_{ij}:

$$S_{ij} = \begin{cases} \sum_p G(\theta_{ij}(p)), \& d_{ij} \leq D_{\max} \\ 0, \text{otherwise} \end{cases} \tag{8}$$

Finally, for each reference image I_{ref_i}, the top 3 source images $I_{\text{src}_i^1}$, $I_{\text{src}_i^2}$, $I_{\text{src}_i^3}$ with the highest matching scores S_{ij} are selected to form the optimal set of views.

Per-point optimal view selection. Our goal is to ensure that each SfM point within a sub-region is associated with its most informative and geometrically robust image pairs. First, we project each point P_k onto the 2D image planes of each reference image I_{ref_i} and corresponding source images $I_{\text{src}_i^1}$, $I_{\text{src}_i^2}$, $I_{\text{src}_i^3}$ that can observe P_k. We then calculate the average euclidean distance $D_{avg}(P_k)$ between each projected point and the corresponding centers. Among all possible four-image groups that observe P_k, we select the

group with the smallest $D_{avg}(P_k)_{min}$. This ensures that P_k is primarily reconstructed using images where it is closest to the image centers in their respective 2D planes, thereby enhancing the reliability of its triangulation.

After determining the best image groups for all 3D points within the sub-region, we exclude any images that do not appear in any of the selected best groups. In our experiments, the excluded images are mainly located at the outermost sides of the sub-region, which means they are redundant images which can only observe a few points in the region.

4.2. Adaptive Multi-View Densification

Prior works [7, 9, 18, 47] have established that finer Gaussian primitives are critical for capturing high-frequency geometric details. In ULSR-GS, we perform additional densification via a MVS-like approach [8], to address the limitations of overly smooth meshes resulting from D_{mean} computations from TSDF fusion. This method projects the D_{mean} into 3D space [8, 42] and combines with the RGB information of the GT image to enrich the Gaussian primitives.

Multi-view depth aggregation. In our approach, we perform weighted averaging of depth information from multiple source views. This weighting assigns a confidence score to each depth estimate based on its geometric consistency [42], ensuring that source views with higher consistency have a greater influence on the final depth estimation.

Scince we have 3 source views from Eq. (8), with each rendered depth D_{mean} denoted as $\mathbf{D}_{\text{src}_i}$ for $i = 1, 2, 3$, and the depth map of the reference view denoted as $\mathbf{D}_{\text{ref}}$. For each pixel p_{ref} in the reference view, the final fused depth estimate $\mathbf{D}_{\text{final}}(p_{\text{ref}})$ is obtained through weighted fusion of

the source views:

$$\mathbf{D}_{\text{final}}(p_{\text{ref}}) = \frac{\sum_{i=1}^{N} w_{\text{src}_i}(p_{\text{src}_i}) \cdot \mathbf{D}_{\text{src}_i}(p_{\text{src}_i})}{\sum_{i=1}^{N} w_{\text{src}_i}(p_{\text{src}_i})}, \quad (9)$$

where $\mathbf{D}_{\text{src}_i}(p_{\text{src}_i})$ is the depth value at the projected pixel p_{src_i} in the i-th source view corresponding to p_{ref}. The weight $w_{\text{src}_i}(p_{\text{src}_i})$ is based on the geometric consistency score, measuring the reliability of the depth estimate from the i-th source view:

$$w_{\text{src}i}(p_{\text{src}i}) = \exp\left(-\frac{E_{\text{depth}}(p_{\text{ref}}, p_{\text{src}i})}{\sigma^2}\right), \quad (10)$$

where $E_{\text{depth}}(\cdot)$ represents the depth error between the reference view and the source view:

$$E_{\text{depth}}(p_{\text{ref}}, p_{\text{src}_i}) = |\mathbf{D}_{\text{ref}}(p_{\text{ref}}) - \mathbf{D}_{\text{src}_i}(p_{\text{src}_i})| \quad (11)$$

Adaptive depth densification. Directly projecting all depth information after checking the geometric consistency introduces excessive redundant information into the training scene [18]. This redundancy negatively impacts both the training speed and the accuracy of the scene representation. To address this issue, we introduce an adaptive densification window mask that confines the densification area. This approach eliminates inaccurate values at the edges of the depth map and adaptively accounts for non-uniform depth changes caused by varying viewpoint poses.

To determine the mask size adaptively, we base it on the depth gradient $|\nabla \mathbf{D}(x)|$, which represents the rate of change in depth at each pixel x. The window size is inversely proportional to the mean gradient $\bar{g}$, implying that regions with higher depth variations have smaller windows to capture finer details, while regions with lower variations have larger windows. We first compute the mean gradient $\bar{g}$ over the entire depth map of size $h \times w$:

$$\bar{g} = \frac{1}{h \times w} \sum_{x \in \mathbb{D}} |\nabla \mathbf{D}(x)| \quad (12)$$

Based on the average gradient $\bar{g}$, we define the height and width of the window to dynamically adapt to the depth changes in the scene:

$$(h_{\text{win}}, w_{\text{win}}) = \frac{k}{\bar{g} + \epsilon} \cdot \frac{(h, w)}{2} \quad (13)$$

where k and ϵ are proportional constants that control the change in window size. The term ϵ prevents division by zero when the gradient is very small.

We then project the depth $\mathbf{D}_{\text{final}}$ within the window $W(u, v) \in (h_{\text{win}}, w_{\text{win}})$ after the geometric consistency check and fusion from Eq. (9):

$$\mathbf{P}_w(u, v) = \mathbf{D}_w(u, v) \cdot \mathbf{K}_p^{-1} \begin{bmatrix} u & v & 1 \end{bmatrix} \quad (14)$$

where $\mathbf{K}_p^{-1}$ is the inverse of the intrinsic camera matrix, and $\mathbf{P}_w(u, v)$ represents the 3D point projected from the windowed depth map. Following the settings of MVG-Splatting [18], we perform additional rescaling and rotational alignment on the newly added Gaussian primitives after each densification step to ensure consistency across the scene.

4.3. Loss Functions

Geometric consistency loss. We optimize the geometric consistency between the reference view and the source views via the *reprojection error* that compares the reprojected reference depth $\mathbf{D}_{\text{ref}}$ with the source depth $\mathbf{D}_{\text{src}}$ in the source view as defined in Eq. (11).

Multi-view normal consistency loss. For each projected point, we compute the angular error between the normal vectors of the reference view $\mathbf{N}_{\text{ref}}$ and the source view $\mathbf{N}_{\text{src}}$. This normal vector consistency ensures that the geometry of the primitive remains consistent under different viewing angles:

$$E_{\text{normal}} = 1 - \frac{\mathbf{N}_{\text{ref}} \cdot \mathbf{N}_{\text{src}}}{|\mathbf{N}_{\text{ref}}||\mathbf{N}_{\text{src}}|} \quad (15)$$

Final loss function. The final loss function is defined as:

$$L = \alpha \cdot E_{\text{depth}} + \beta \cdot E_{\text{normal}} + L_{\text{geo}} + L_r \quad (16)$$

where α, β, are weighting factors that balance the contributions of each error term. L_{geo} includes two regularization components from 2DGS [11]: depth distortion ℓ_d and normal consistency ℓ_n. L_r represents the RGB reconstruction loss, combining L_1 loss and the D-SSIM metric as used in 3DGS [12].

5. Experiments

The comprehensive experimental setups are described in Section 5.1. Assessments of mesh and rendering quality are provided in Section 5.2. Finally, in Section 5.3, we analyze the proposed partition strategy and conduct ablation studies.

5.1. Experimental Setup.

Implementation. In our experiments, we initially employ COLMAP [29] to obtain the SfM results for the entire scene. Subsequently, we align the sparse point cloud using the Manhattan World Alignment, ensuring that the *y-axis* is perpendicular to the *xz* plane. We set the angular threshold $\theta_{min} = 90$, and the distance threshold $D_{max} = \sqrt{d_x \times d_y}$, where d_x and d_y are the distance of each sub-region. After partitioning, we double the boundary of the initial point cloud for each region for the initialization. This configuration ensures that the edges of each subregion receive sufficient training information, thereby preventing edge mismatches during the stitching process. Redundant points are automatically pruned during training through opacity

Scene	LOWER_CAMP			CUHK_UPPER			LFLS			SMBU			SZIIT			SZTU			
Metrics ($\tau = 0.025$)	Pre.↑	Rec.↑	F1↑	Pre.↑	Rec.↑	F1↑	Pre.↑	Rec.↑	F1↑	Pre.↑	Rec.↑	F1↑	Pre.↑	Rec.↑	F1↑	Pre.↑	Rec.↑	F1↑	Time
Neuralangelo	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	>100 h
SuGaR	0.279	0.263	0.271	0.261	0.273	0.267	0.271	0.300	0.285	0.376	0.409	0.392	0.249	0.276	0.263	0.272	0.299	0.285	4.1h
2DGS-60K	0.583	0.693	0.633	0.554	0.563	0.559	0.543	0.557	0.550	0.580	0.745	0.652	0.584	0.617	0.600	0.596	0.613	0.604	2.1h
GOF-60K	0.667	0.652	0.659	0.693	0.672	0.682	0.607	0.659	0.631	0.679	0.784	0.727	0.649	0.695	0.671	-	-	-	5.3h
PGSR-60K	0.683	0.648	0.665	0.672	0.673	0.672	-	-	-	0.660	0.795	0.721	0.679	0.642	0.659	0.548	0.542	0.545	5.7h
Ours (4 GPU)	0.681	0.689	0.685	0.690	0.705	0.697	0.684	0.703	0.693	0.737	0.789	0.762	0.705	0.713	0.709	0.713	0.723	0.718	3.7h

Table 1. **Quantitative results on GauU-Scene dataset [38].** We report precision, recall, and F-1 score at the threshold $\tau = 0.025$. The best, second best, and third best results are highlighted in red , orange , and yellow respectively. "-" denotes no results due to GPU out of memory. " *" denotes no result due to no convergence.

Scene	Building			Rubble			Campus			Residence			Sci-Art		
Metrics	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Mega-NeRF	21.48	0.569	0.378	24.70	0.575	0.407	23.93	0.561	0.513	22.86	0.648	0.330	26.25	0.769	0.263
Switch-NeRF	22.07	0.594	0.332	24.93	0.586	0.377	24.03	0.565	0.495	23.41	0.675	0.280	27.07	0.793	0.224
3DGS-60K	22.65	0.742	0.163	26.32	0.786	0.179	23.47	0.703	0.298	23.21	0.791	0.152	24.99	0.805	0.172
2DGS-60K	22.25	0.739	0.198	26.70	0.785	0.204	23.20	0.632	0.342	22.35	0.762	0.162	24.68	0.811	0.219
Mip-Splatting-60K	22.78	0.750	0.160	26.35	0.780	0.172	-	-	-	23.44	0.785	0.150	24.96	0.810	0.175
Vast Gaussian	23.50	0.804	0.130	26.92	0.823	0.132	26.00	0.816	0.151	24.25	0.852	0.124	26.81	0.885	0.121
Vast Gaussian[†]	23.45	0.780	0.152	26.91	0.805	0.148	25.98	0.793	0.163	24.21	0.822	0.137	26.75	0.878	0.138
ULSR-GS (Ours)	23.49	0.812	0.142	26.95	0.813	0.140	25.99	0.814	0.146	24.24	0.847	0.138	26.81	0.895	0.109

Table 2. **Novel view synthesis results on Mill19 [32] and UrbanScene [21] 3D Dataset.** The best, second best, and third best results are highlighted in red , orange , and yellow respectively. Vast Gaussian[†] denotes an unofficial reproduction of Vast Gaussian [19]. "-" denotes no results due to GPU out of memory.

culling. Experiments for our method are conducted on 4 RTX 4090 GPUs, and each sub-region is trained individually by one GPU.

For each subregion, training is conducted for 40k iterations. Adaptive multi-view densification begin at the 20k-th iteration and terminates at the 30k-th iteration. Regarding the loss function, we set the parameters $\alpha = 0.01$ and $\beta = 0.1$. All other loss functions and training parameters are configured identically to those used in 3DGS and 2DGS frameworks. After training, we employ TSDF [35] to extract the mesh of each sub-region then crop using their partition bounding boxes and subsequently stitched together.

Datasets and metrics. Our experiments are conducted on large-scale aerial oblique photogrammetry dataset: GauU-Scene (GauU) [38], Mill 19 [32] and UrbanScene3D (US3D)[21]. For evaluating NVS rendering results, we employ PSNR, SSIM, and LPIPS on US3D and Mill 19 datasets. In assessing the reconstructed meshes, we report Precision, Recall, and F-score on GauU-scene dataset. Following the settings of Vast Gaussian [19], we adjust the training configurations for all GS-based methods for a fair comparison.

Compared methods. We compare our proposed ULSR-GS with GS-based methods capable of mesh extraction, including SuGaR [10], 2DGS [11], GOF [47], and PGSR [3]. For NVS rendering evaluation, we include NeRF-based meth-
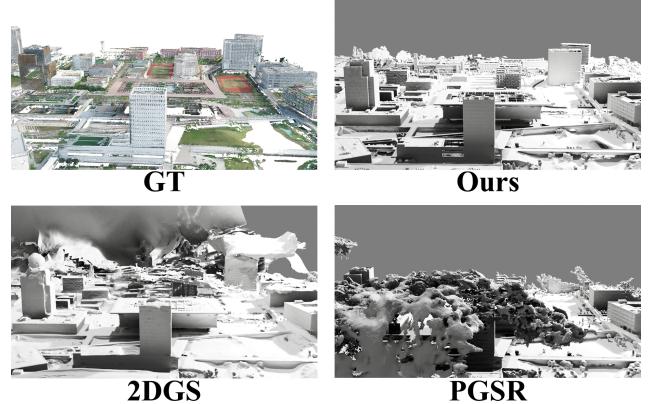

GT   Ours

2DGS   PGSR

Figure 3. **Qualitative comparison on the GauU-Scene [21]dataset *SZTU* scene.** When the scene range expands and the number of images increases, the geometric reconstruction quality of 2DGS [11] and PGSR [3] drops sharply.

ods Mega-NeRF [32] and Switch-NeRF [25], as well as the GS-based 3DGS [12], Mip-Splatting [46] and Vast Gaussian [19]. Since Vast Gaussian is not open-sourced, we report both the results from the original publication and a GitHub unofficial implementation result.
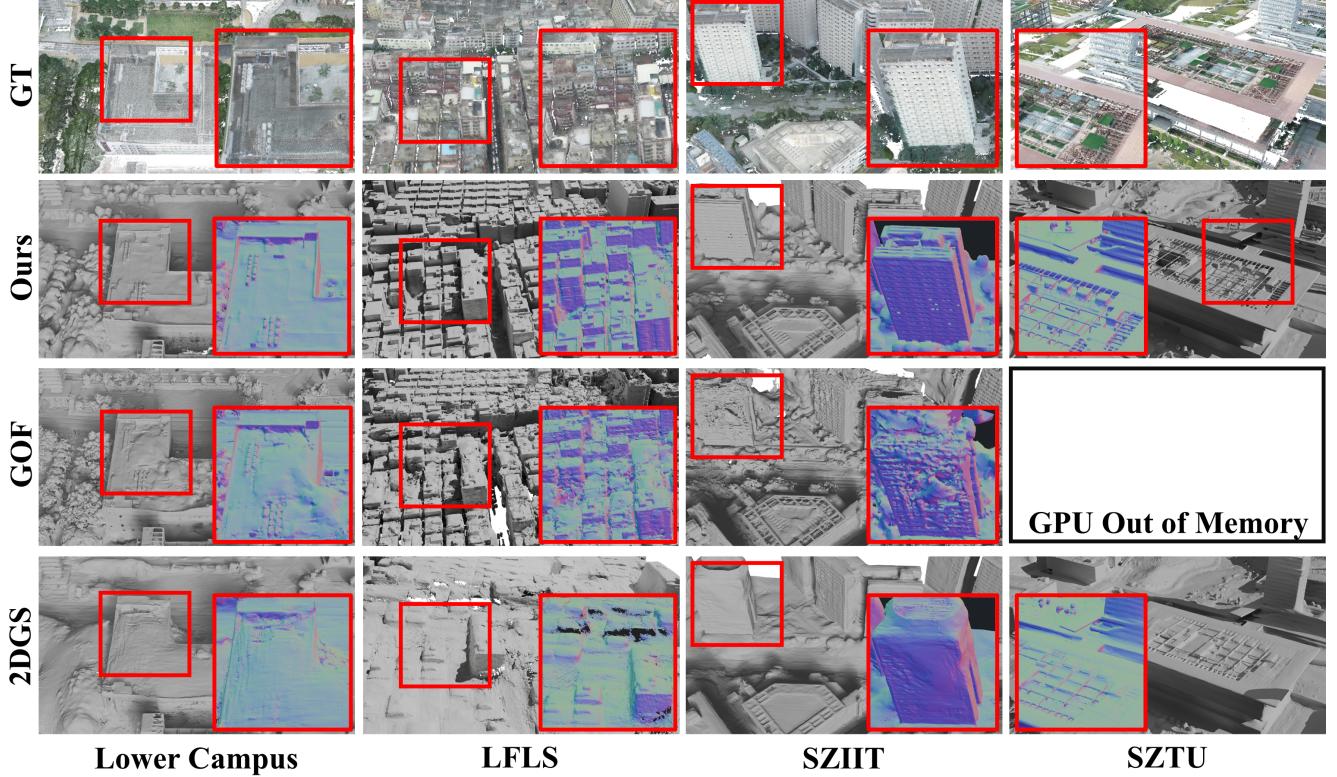
Figure 4. **Qualitative comparison on the GauU-Scene dataset [38].** We demonstrate the zoomed-in surface normal detail of the selected areas in red box.

Figure 5. **Qualitative comparison of rendering on the UrbanScene 3D [21] dataset.** We demonstrate the zoomed-in surface normal detail of the selected areas in red box. VastGaussian [†] means the unofficial reproduction of Vast Gaussian [19]

5.2. Results

Mesh reconstruction results. Table 1 presents a comparison of reconstruction quality on the GauU-Scene dataset [38]. Our method significantly outperforms other GS-based approaches. When the number of images in the scene (*LFLS*, *SZIIT* (see Figure 3), and *SZTU*) exceeds 1000, the reconstruction quality indicators of 2DGS [11], PGSR [3] and GOF [47] are far behind us. In terms of average training duration, our method exhibits a notably shorter total training time on four GPUs compared to single GPU methods PGSR [3] and GOF [47] at 60K iterations. It is worth mentioning that GOF [47] and PGSR [3] employ similar densification methods, which result in GPU out-of-memory (OOM) issues in certain scenes. This further underscores the advantages of our partitioning strategy. Figure 4 presents sev-

eral examples of reconstructed meshes, illustrating that our method exceeds existing approaches in both the richness of details and precision. Detailed qualitative and quantitative comparisons are summarised in the supplementary material.
Rendering results. In Table 2, we compare the performance of our proposed ULSR-GS method with various main approaches in different scenes. Although ULSR-GS is designed primarily for precise surface extraction, it also demonstrates exceptional performance on NVS metrics, and Figure 5 presents an representative rendering comparisons.
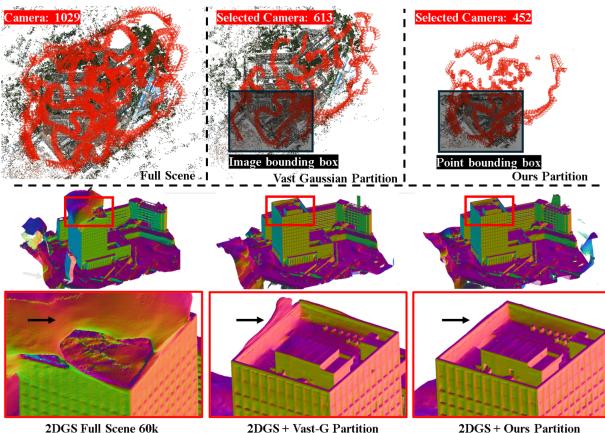
5.3. Analysis

Point partition. Table 3 presents comparison experiments for partition strategies. Without pair selection (Table 3 (A)),

an excessive number of images participate in the training of each region, adversely affecting both the training time and the quality of mesh extraction. These redundant images are often caused by errors in the SfM calculations [29]. To further demonstrate the feasibility of our partitioning strategy, we train ULSR-GS and the baseline method 2DGS [11] on our partition strategies and the Vast Gaussian's [19] (Table 3 (B), (C) and (D)). The results show that our partition strategy is superior to the image position-based approach in processing large-scale surface extraction tasks. Figure 6 shows a visual comparison on a more challenging close-range dataset with specific path planning [21]. Our method selects a greater number of relevant training images within a smaller area, while the Vast Gaussian [19] strategy selects a wider range, resulting in a lack of information in the partition area. More results can be found in the Supplementary Material.

GauU-Scene (Avg. images: 957)	Pre. ↑	Rec. ↑	F1 ↑	Avg. Image	Time ↓
A. w/o sel. (4 GPU)	0.699	0.707	0.702	437	4.5 h
B. Ours + VastG (4 GPU)	0.685	0.700	0.692	359	3.5 h
C. 2DGS + Ours PP (4 GPU)	0.677	0.671	0.674	366	3.0 h
D. 2DGS + VastG † PP (4 GPU)	0.663	0.669	0.667	359	2.9 h
E. 2DGS-60K (1 GPU)	0.573	0.631	0.600	957	2.1 h
F. ULSR-GS (4 GPU)	0.713	0.725	0.717	366	3.7 h

Table 3. **Partition Strategies Comparison.** We conduct experiments on the GauU-Scene dataset [38] scene in 4×4 cells. We reported metrics including Precision, Recall, F-1 Score, Average number of images per cell, and Total training time. VastG † PP means the unofficial reproduction partition method of Vast Gaussian [19].
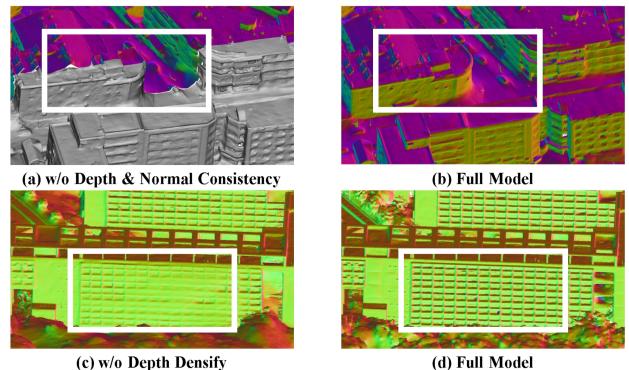

Figure 6. **Qualitative comparison of different partition strategy on close-range photogrammetry dataset.** We train 2DGS [11] as baseline method on UrbanScene 3D [21] *artsci_corase* scene.

Ablations. Table 4 showcases ablation experiments for the proposed geometric consistency constraints. Specifically, in Table (4 (A), (B) and (C)) the absence of multi-view depth

consistency and normal consistency leads to a decline in mesh quality. We compare densification in Tables 4 (D) and (E), where if we cancel the window mask, the scene will have too many Gaussian points, resulting in OOM. As illustrated in Figure 7, without incorporating E_{depth} and E_{normal}, the quality of surface extraction deteriorates, and the edges of each region fail to stitch seamlessly. And removing depth densify results in excessively smooth mesh extraction outcomes.

LOWER_CAMPUS	Precision. ↑	Reccall. ↑	F-1 Score ↑
A. w/o E_{depth} & E_{normal}	0.659	0.661	0.660
B. w/o E_{depth}	0.663	0.665	0.664
C. w/o E_{normal}	0.652	0.651	0.651
D. w/o densify	0.678	0.685	0.682
E. w densify & w/o window mask	-	-	-
F. ULSR-GS	0.681	0.689	0.685

Table 4. **Geometric Consistency Constraints Ablation Studies.** We conducte ablation experiments on the GauU-Scene dataset [38] *LOWER_CAMPUS* scene in 4×4 cells. We reported metrics including Precision, Recall, F-1 Score. "-" denotes no data due to GPU out of memory.

Figure 7. **Qualitative comparison on the geometric consistency constraints ablation studies.** Without E_{depth} & E_{normal} edges of each region fail to stitch seamlessly; Without adaptive densify results in excessively smooth mesh extraction outcomes.

6. Conclusion

We present ULSR-GS, a framework dedicated to high-fidelity surface extraction in ultra-large-scale scene. Specifically, our partitioning approach combines with a multi-view selection strategy. Additionally, ULSR-GS employs a multi-view geometric consistency densification to enhance surface details. Experimental results demonstrate that ULSR-GS outperforms other SOTA GS-based works on large-scale benchmark datasets.

ULSR-GS exhibits some limitations when reconstructing highly reflective areas, such as water surfaces and glass

buildings. Additionally, despite accurately recalculating rendered depth, computational errors are prone to occur in certain occluded regions. Therefore, future research should explore solutions to address these challenges.

References

[1] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021. 2

[2] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, pages 1–11, 2011. 2

[3] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient high-fidelity surface reconstruction. 2024. 1, 3, 6, 7, 4, 5

[4] Junyi Chen, Weicai Ye, Yifan Wang, Danpeng Chen, Di Huang, Wanli Ouyang, Guofeng Zhang, Yu Qiao, and Tong He. Gigags: Scaling up planar-based 3d gaussians for large scene surface reconstruction, 2024. 2

[5] Mingfei Chen, Jianfeng Zhang, Xiangyu Xu, Lijuan Liu, Yujun Cai, Jiashi Feng, and Shuicheng Yan. Geometry-guided progressive nerf for generalizable and efficient neural human rendering. In *ECCV*, 2022. 2

[6] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1538–1547, 2019. 3

[7] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*, 2024. 2, 4

[8] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. In *Forty-first International Conference on Machine Learning*, 2024. 2, 4

[9] Lue Fan, Yuxue Yang, Minxing Li, Hongsheng Li, and Zhaoxiang Zhang. Trim 3d gaussian splatting for accurate geometry representation. *arXiv preprint arXiv:2406.07499*, 2024. 3, 4

[10] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023. 1, 2, 3, 6, 4, 7

[11] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. 1, 3, 5, 6, 7, 8, 4

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time

radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2, 3, 5, 6

[13] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics*, 43(4), 2024. 2, 4

[14] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 3

[15] Itai Lang, Asaf Manor, and Shai Avidan. SampleNet: Differentiable Point Cloud Sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7578–7588, 2020. 3

[16] Yixuan Li, Lihan Jiang, Linning Xu, Yuanbo Xiangli, Zhenzhi Wang, Dahua Lin, and Bo Dai. Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. *arXiv e-prints*, pages arXiv–2308, 2023. 3

[17] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2023. 2

[18] Zhuoxiao Li, Shanliang Yao, Yijie Chu, Angel F. Garcia-Fernandez, Yong Yue, Eng Gee Lim, and Xiaohui Zhu. Mvg-splatting: Multi-view guided gaussian splatting with adaptive quantile-based geometric consistency densification, 2024. 1, 2, 3, 4, 5

[19] Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, and Wenming Yang. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5166–5175, 2024. 1, 2, 3, 6, 7, 8, 4

[20] Jin Liu, Jian Gao, Shunping Ji, Chang Zeng, Shaoyi Zhang, and JianYa Gong. Deep learning based multi-view stereo matching and 3d scene reconstruction from oblique aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 204:42–60, 2023. 2

[21] Yilin Liu, Fuyou Xue, and Hui Huang. Urbanscene3d: A large scale urban scene dataset and simulator. 2021. 1, 6, 7, 8, 2, 3

[22] Yang Liu, He Guan, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. *arXiv preprint arXiv:2404.01133*, 2024. 1, 2, 3, 4

[23] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998. 2

[24] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 1, 2

[25] Zhenxing Mi and Dan Xu. Switch-nerf: Learning scene decomposition with mixture of experts for large-scale neural radiance fields. In *International Conference on Learning Representations (ICLR)*, 2023. 2, 6

[26] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2

[27] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. 2021. 2

[28] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001. 3

[29] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 3, 4, 5, 8

[30] Wanjuan Su and Wenbing Tao. Efficient edge-preserving multi-view stereo network for depth estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2348–2356, 2023. 3

[31] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. *arXiv*, 2022. 2

[32] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12922–12931, 2022. 2, 6

[33] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. Dn-splatter: Depth and normal priors for gaussian splatting and meshing, 2024. 3

[34] 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. *NeurIPS*, 2021. 2

[35] Diana Werner, Ayoub Al-Hamadi, and Philipp Werner. Truncated signed distance function: experiments on voxel size. In *Image Analysis and Recognition: 11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22-24, 2014, Proceedings, Part II 11*, pages 357–364. Springer, 2014. 3, 6

[36] Jiang Wu, Rui Li, Haofei Xu, Wenxun Zhao, Yu Zhu, Jinqiu Sun, and Yanning Zhang. Gomvs: Geometrically consistent cost aggregation for multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20207–20216, 2024. 2, 3

[37] 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 *European conference on computer vision*, pages 106–122. Springer, 2022. 2

[38] Butian Xiong, Nanjun Zheng, Junhua Liu, and Zhen Li. Gauu-scene v2: Assessing the reliability of image-based metrics with expansive lidar image dataset using 3dgs and nerf, 2024. 6, 7, 8, 1, 3, 4, 5

[39] Kaiqiang Xiong, Rui Peng, Zhe Zhang, Tianxing Feng, Jianbo Jiao, Feng Gao, and Ronggang Wang. Cl-mvsnet: Unsupervised multi-view stereo with dual-level contrastive learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3769–3780, 2023. 2

[40] 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, 2023. 2

[41] Qingshan Xu, Weihang Kong, Wenbing Tao, and Marc Pollefeys. Multi-scale geometric consistency guided and planar prior assisted multi-view stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4945–4963, 2022. 2

[42] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783, 2018. 2, 3, 4

[43] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *NeurIPS*, 2021. 2

[44] 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. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023. 2

[45] Zehao Yu and Shenghua Gao. Fast-mvsnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1949–1958, 2020. 2, 3

[46] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 2, 6

[47] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient high-quality compact surface reconstruction in unbounded scenes. *arXiv:2404.10772*, 2024. 3, 4, 6, 7, 5

[48] Gim Hee Lee Yu Chen. Dogaussian: Distributed-oriented gaussian splatting for large-scale 3d reconstruction via gaussian consensus. In *arXiv*, 2024. 1, 2, 3

[49] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024. 1, 3

[50] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2
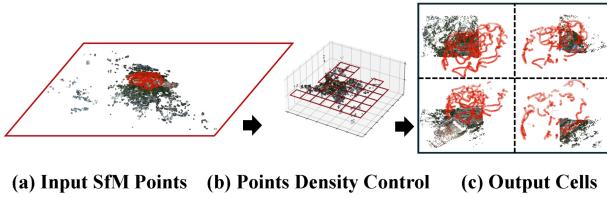
ULSR-GS: Ultra Large-scale Surface Reconstruction Gaussian Splatting with Multi-View Geometric Consistency

Supplementary Material

7. Detailed Partition Implementations and Comparisons

7.1. Additional implementation details

In this section, we present a description of our partitioning method. As illustrated in Figure 8, we optimize the partition process by focusing on the most relevant parts of the scene.

(a) Input SfM Points   **(b) Points Density Control**   **(c) Output Cells**

Figure 8. **Density-controlled point cloud filtering for boundary refinement.** We demonstrate a case of using density control to filter out sparse point clouds around reconstructed targets from precisely controlled point cloud boundaries in the UrbanScene3D [21] dataset *artsci_coarse* scene. (a) In the input SfM point cloud, the incorrect matching points outside the reconstructed scene may cause the boundary to become larger. (b) We divide the scene into countless small regions on the *xz* plane and evaluate the point cloud density of the region, throw out regions that are too low in density. (c) The boundary is significantly reduced by density-controlled SfM point clouds, allowing for fine-grained partition tasks.

Detailed per-point view selection. For each point $P_k \in \mathcal{P}$, we consider all possible groups of images that observe P_k. Specifically, each group $\mathcal{G}_i = I_{\text{ref}_i}, I_{\text{src}_i^1}, I_{\text{src}_i^2}, I_{\text{src}_i^3}$ consists of one reference image and three corresponding source images. Our method involves the basic *W2C* steps. We project P_k onto the 2D image planes of each image $I_j \in \mathcal{G}_i$:

$$\begin{bmatrix} u_{k_j} \\ v_{k_j} \\ 1 \end{bmatrix} = \frac{1}{Z_c} \begin{bmatrix} f_{x_j} & 0 & c_{x_j} & 0 \\ 0 & f_{y_j} & c_{y_j} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_j & t_j \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X_{k_j} \\ Y_{k_j} \\ Z_{k_j} \\ 1 \end{bmatrix}$$
(17)

where (u_{k_j}, v_{k_j}) are the normalized pixel coordinates of I_j, f_{x_j} and f_{x_j} are the focal lengths of camera I_j along the x and y axes, respectively. c_{x_j} and c_{y_j} are the image center coordinates. $R_j \in SO(3)$ and t_j are the rotation matrix and translation vector of I_j respectively. We then compute the Euclidean distance between the projected point (u_{k_j}, v_{k_j})

and the principal point (image center) (c_{x_j}, c_{y_j}) of image I_j:

$$d_{k_j} = \sqrt{(u_{k_j} - c_{x_j})^2 + (v_{k_j} - c_{y_j})^2}.$$
(18)

For each image group $\mathcal{G}_i$ observing P_k, we we calculate the average Euclidean distance:

$$D_{avg}(P_k, \mathcal{G}_i) = \frac{1}{|\mathcal{G}_i|} \sum_{I_j \in \mathcal{G}_i} d_{k_j}$$
(19)

Since we want to select the optimal image group for every initial SfM point in the sub-regions. We simply identify the group for P_k $\mathcal{G}$ by minimizing D_{avg}:

$$\mathcal{G}_{opt}(P_k) = agr \min_{\mathcal{G}_i} D_{avg}(P_k, \mathcal{G}_i).$$
(20)

This ensures that P_k is reconstructed using images where it appears closest to the image centers, enhancing reconstruction reliability due to reduced distortion and higher image quality near the center. After determining $\mathcal{G}_{\text{opt}}(P_k)$ for all points $P_k \in \mathcal{P}$, we aggregate the utilized images I_{used} and exclude any images not present:

$$I_{used} = \bigcup_{P_k \in \mathcal{P}} \mathcal{G}_{opt}(P_k)$$
(21)

Sub-regions with insufficient data To address the issue where some sub-regions may contain too few points (especially in boundary areas) after dividing the scene into smaller parts, we evaluate each sub-region as follows: We retain a sub-region only if both the number of points and the number of matched images within it reach at least 10 % of the average per-subregion counts (i.e., the total number of points or images divided by n^2).

Scene	Images	Grid Size	Used
Campus [21]	5871	8×8	41
Residence [21]	2582	6×6	20
SZTU [38]	1500	6×6	17
LOWER_CAMP [38]	670	4×4	12

Table 5. **Example Per Scene Partitioning Strategy.**

Scene Partitioning Strategy We partition each scene based on the number of images to ensure efficient processing and optimal reconstruction quality. Specifically, we divide the regions according to the following criteria:

Figure 9. **Qualitative comparison of partition strategies.** In close-range photography tasks, a large number of background areas will appear. Our partitioning method emphasizes high-quality rendering of the reconstructed subject and ignores the division of background areas.

1. Scenes with fewer than 1,000 images are divided into a 4×4 grid.
2. Scenes with fewer than 3,000 images are divided into a 6×6 grid.
3. Scenes with more than 3,000 images are divided into an 8×8 grid.

Table 5 lists the examples of the number of initially partitioned sub-regions for each scene and the number of sub-regions that were ultimately included in the training after applying our filtering criteria based on point cloud density and image coverage.

7.2. Additional partition comparisons

Novel view synthesis. Table 6 quantitatively compares different partitioning methods, specifically our point-based partitioning approach and the image-based method employed by Vast Gaussian [19]. The results indicate that the Vast Gaussian partitioning strategy is more suitable for rendering tasks. This is because our method prioritizes the reconstruction of the scene's primary structure by excluding the distant sparse SfM points during sub-region partitioning. Figure 9 provides a qualitative comparison to further illustrate this difference. When training ULSR-GS using different partitioning strategies, our partition method shows a trade-off: while it loses performance in reconstructing the background, it preserves significantly more detail in the reconstruction of the primary scene components.

Mesh extraction. Since UrbanScene 3D [21] does not provide a reference ground truth point cloud, we perform a qualitative analysis. And Figures 10 and 11 qualitatively demonstrate comparisons on both five-directional oblique photogrammetry datasets and close-range oblique photogrammetry datasets. Our partitioning strategy effectively identifies and selects the most critical images for the reconstruction regions, significantly enhancing the detail and accuracy of the extracted meshes.

Figure 12 demonstrates a qualitative analysis of partition
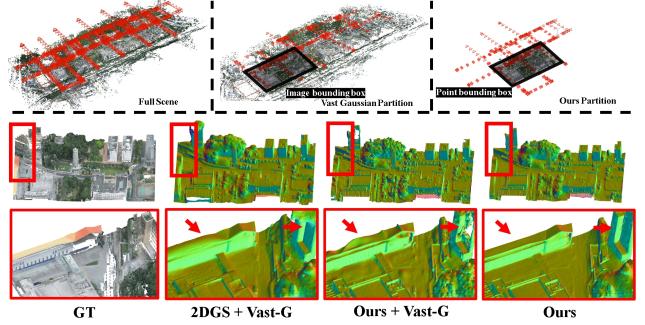

Figure 10. **Qualitative comparison of partition strategies.**
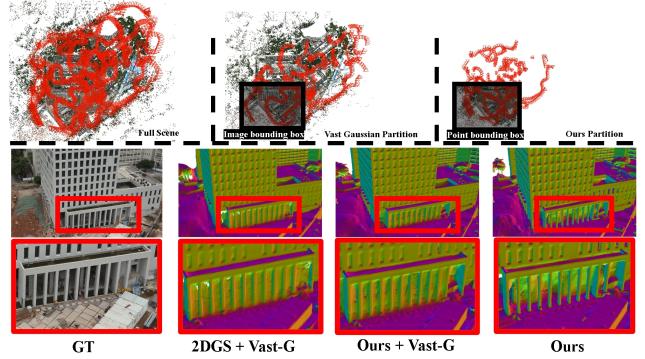

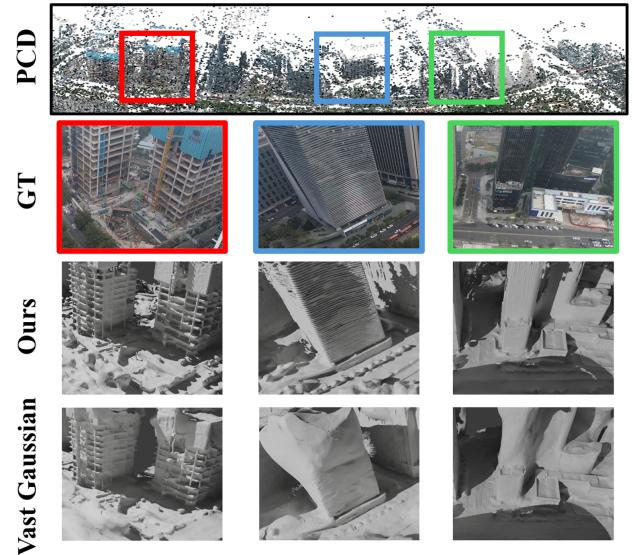Figure 11. **Qualitative comparison of partition strategies.**

Figure 12. **Qualitative comparison of partition strategies at scene edges.** We train our method on our partition strategy and Vast Gaussian's [19] in 8×8 cells. From top to bottom: example scene edge, example GT images at the scene edge, ULSR-GS on our partition strategy, and ULSR-GS on Vast Gaussian's [19] partition strategy.

Data Type	Five-directional			Close-range path planning					
Scene	**CUHK_LOWER [38]**			**Artsci [21]**			**Polytec [21]**		
Metrics	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
2DGS	23.85	0.711	0.382	25.42	0.763	0.346	24.96	0.849	0.236
2DGS+Vast Gaussian	24.98	0.801	0.247	26.98	0.845	0.299	26.31	0.860	0.221
2DGS+Ours Partition	24.87	0.789	0.250	26.72	0.826	0.312	26.02	0.855	0.229
Ours+Vast Gaussian	25.21	0.805	0.234	27.14	0.883	0.280	27.11	0.869	0.207
Ours	25.21	0.810	0.239	27.01	0.851	0.289	26.97	0.861	0.213

Table 6. **Additional partition comparison.** We conduct comparative experiments on different types of oblique photography datasets. Our partition strategy overemphasizes the reconstruction of the main area and deletes the background area with only a few reconstructed images, resulting in a decrease in the overall rendering quality.

strategies at the edges of reconstructed scenes. Comparison was performed on the *campus* scene from the UrbanScene3D [21] dataset. Our partition strategy demonstrates the ability to provide sufficient training images even at the periphery of the scene. In cases of uneven camera distribution (particularly at the periphery of the scene), the image-based partitioning may lead to insufficient coverage of edge regions.

8. Detailed Additional Densification Implementations

8.1. Implementations

Our depth projection process is integrated into the training pipeline after the default densification phase of the vanilla 2DGS [11]. The densification process employs geometric consistency checks with thresholds set to *geo_pixel_thres* = 1 and *geo_depth_thres* = 0.001 to ensure robust depth projection. We set the proportional constant of the window size to k=0.6 in our experiments.

The densification begins at the 20,000th iteration to ensure that accurate rendered depths are available prior to projection. During each densification iteration, we perform local scale re-initialization within the projection area to better adapt to the scene's depth variations. To balance computational efficiency and model performance, we reduce the interval of opacity culling operations to 2,000 iterations. This adjustment prevents over-fitting by limiting the excessive accumulation of projected points while maintaining sufficient pruning to minimize computational overhead.

9. Addition Results

9.1. Mesh evaluation process

Since GauU-scene [38] not provide a standard evaluation code, we follow the Tanks and Tample [14] evaluation pipeline to quantitatively assess reconstructed meshes against GT LiDAR point clouds:

	$\tau = 0.01$			$\tau = 0.005$		
Metrics	Pre.↑	Rec.↑	F1↑	Pre.↑	Rec.↑	F1↑
Neuralangelo	*	*	*	*	*	*
SuGaR	0.246	0.243	0.244	0.212	0.209	0.210
2DGS-60K	0.403	0.432	0.416	0.235	0.267	0.250
GOF-60K	0.429	0.451	0.439	0.262	0.273	0.267
PGSR-60K	0.417	0.429	0.422	0.249	0.253	0.250
Ours (Median depth)	0.432	0.461	0.447	0.262	0.271	0.266
Ours	0.429	0.467	0.451	0.261	0.280	0.269

Table 7. **Quantitative results on GauU-Scene dataset [38].** We report precision, recall, and F-1 score at the threshold $\tau = 0.01, 0.005$. The best, second best, and third best results are highlighted in red, orange, and yellow respectively. "-" denotes no results due to GPU out of memory. "*" denotes no result due to no convergence.

1. To focus the evaluation on relevant regions, we compute the overlapping axis-aligned bounding box of both point clouds and mesh and crop them accordingly. We then perform Iterative Closest Point [28] (ICP) alignment to minimize misalignment between the reconstructed and GT point clouds.
2. We uniformly sample number of points from the reconstructed mesh and GT points to 10 million.
3. We utilize *KDTreeFlann* to compute precision, recall, and F-score using predefined distance thresholds ($\tau = 0.025, 0.01, 0.005$).

9.2. Additional mesh results

To further evaluate the reconstruction accuracy, we conducted experiments on the GauU-Scene dataset [38] using stricter thresholds of $\tau = 0.01$ and $\tau = 0.005$ as presented in Table 7. Our method consistently outperforms existing approaches across both thresholds. At $\tau = 0.01$, our method achieves the highest precision and recall, resulting in the best F-score.

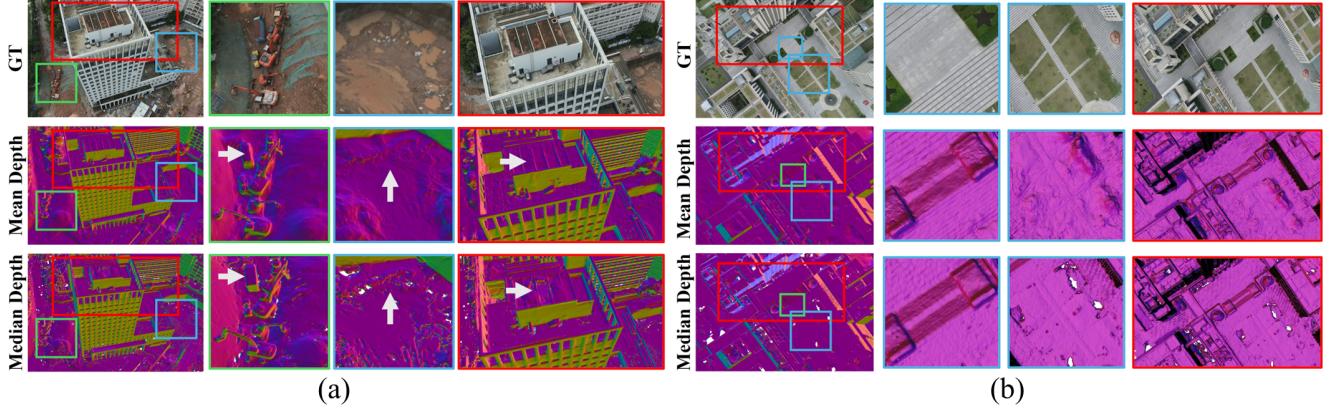Notably, as the evaluation threshold τ becomes more

Figure 13. **Qualitative comparisons between two depth calculations.** Although using median depth in our method can improve the geometry of the building, a large number of holes will appear in the plane area.

stringent (from 0.01 to 0.005), the performance metrics of all models tend to converge, indicating increased difficulty in achieving high precision and recall. Despite this convergence, our method maintains a leading edge, achieving the highest recall and F1 score at $\tau = 0.005$.

9.3. Depth comparison

Table 7 shows the comparison results of mesh extraction using two different methods. Using median depth can further improve the precision of the mesh, but the overall index has decreased, which leads to a decrease in the overall F1 score. Figure 13 further reveals this result, which can be seen that although using median depth can visually improve the fineness of the mesh, it results in depth calculation errors for some smooth objects, such as the ground, resulting in a large number of holes in the mesh.

9.4. Additional qualitative results

Examples of the partitioning mesh results can be found in Figure 19. A full scene comparison on the GauU-Scene [38] dataset among our method, 2DGS [11], GOF[47], and PGSR[3] can be found in Figure 14. Detailed qualitative comparisons between 2DGS [11], GOF[47], PGSR[3], and SuGaR [10] are provided in Figure 15, 16, 17, and 18, respectively.

10. Detailed Conclusion and Limitations

Partition. Our approach addresses key limitations of image-based partitioning methods [13, 19, 22] by ensuring finer granularity in mesh boundaries and prioritizing the reconstruction of key scene regions. This strategy enables us to extract meshes individually rather than merge sub-regions and extract the full scene. Through both quantitative and qualitative evaluations, we demonstrated significant improvements in reconstruction accuracy and detail

preservation, particularly in close-range path planning photogrammetry datasets.

However, our method exhibits certain limitations. Specifically, the emphasis on prioritizing the reconstruction of key areas within the scene inadvertently diminishes the fidelity of distant background regions. While this approach is advantageous for tasks like mesh extraction that focus on scene details, it results in suboptimal performance for rendering tasks that demand comprehensive scene coverage, including distant and sparsely observed areas. Future work can focus on addressing these limitations by extending the partition strategy to achieve a balance between reconstruction fidelity in critical regions and the preservation of distant background elements.

Depth Projection Densification. Our depth projection densification strategy primarily targets improving the quality of well-reconstructed regions, enhancing their rendering and reconstruction fidelity. However, it does not address under-reconstructed areas effectively. Additionally, the reliance on multi-view geometric consistency constraints can lead to instability in reflective regions, such as water surfaces and glass. These limitations highlight the need for future work to develop more robust densification and constraints strategies that not only address under-reconstructed areas but also improve the handling of challenging reflective surfaces.
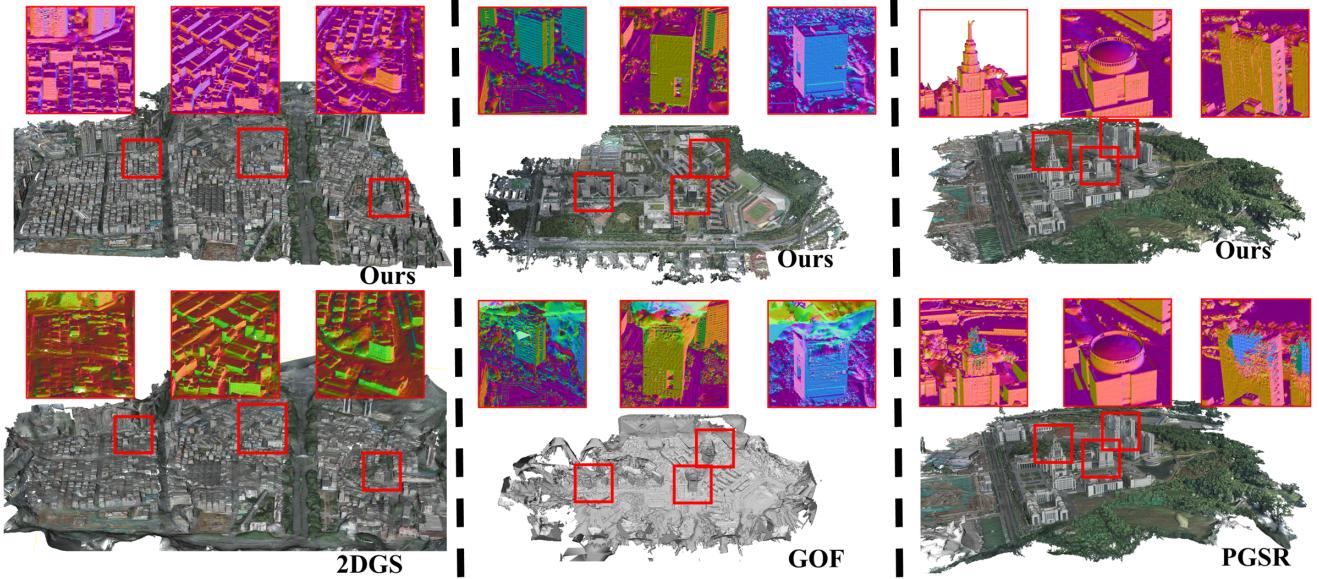
4

Figure 14. **Full scene comparison.** We show the full scene comparison with 2DGS [11], GOF[47], and PGSR[3]. From left to right are the *LFLS*, *SZIIT*, and *SMBU* scenes of the GauU-Scene [38] dataset.
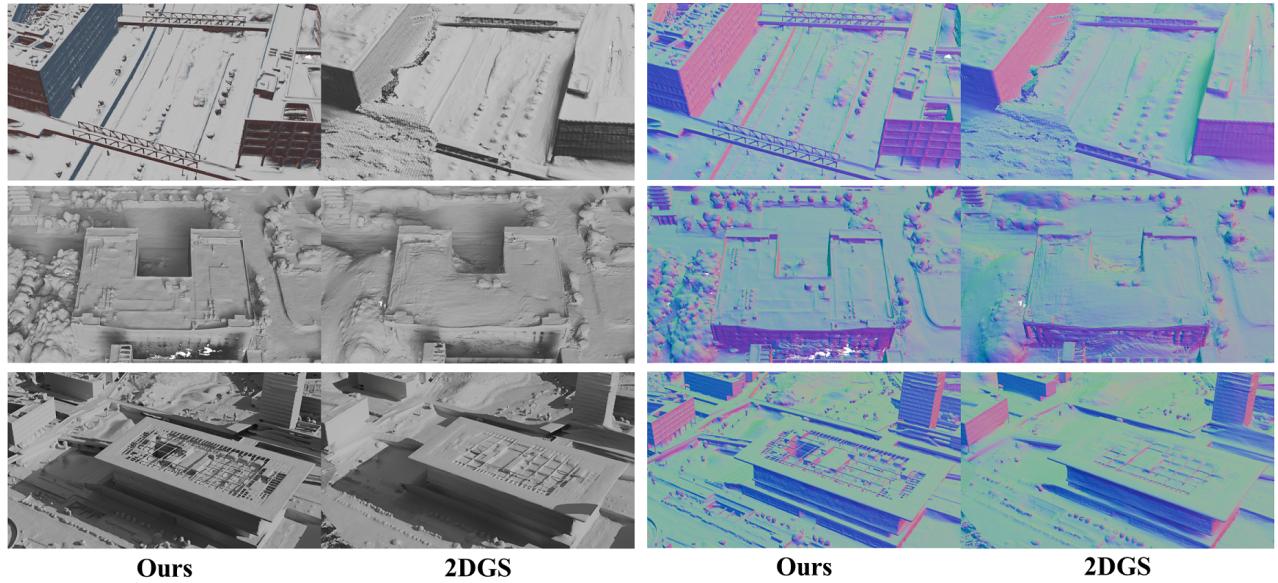

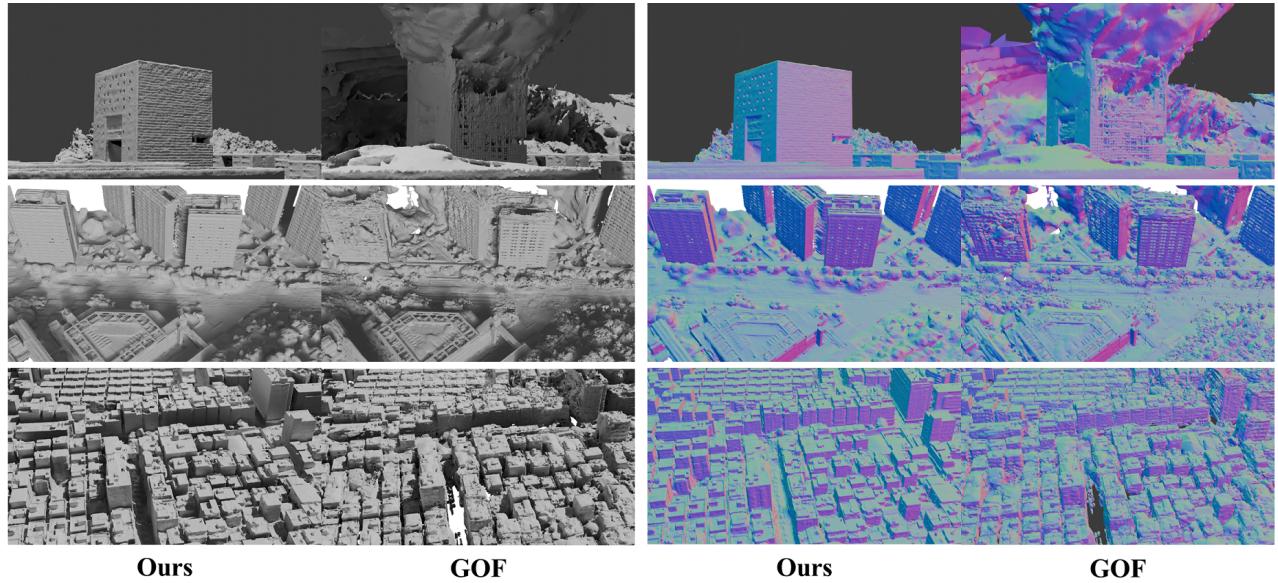Figure 15. Qualitative comparisons between ours and 2DGS [11].

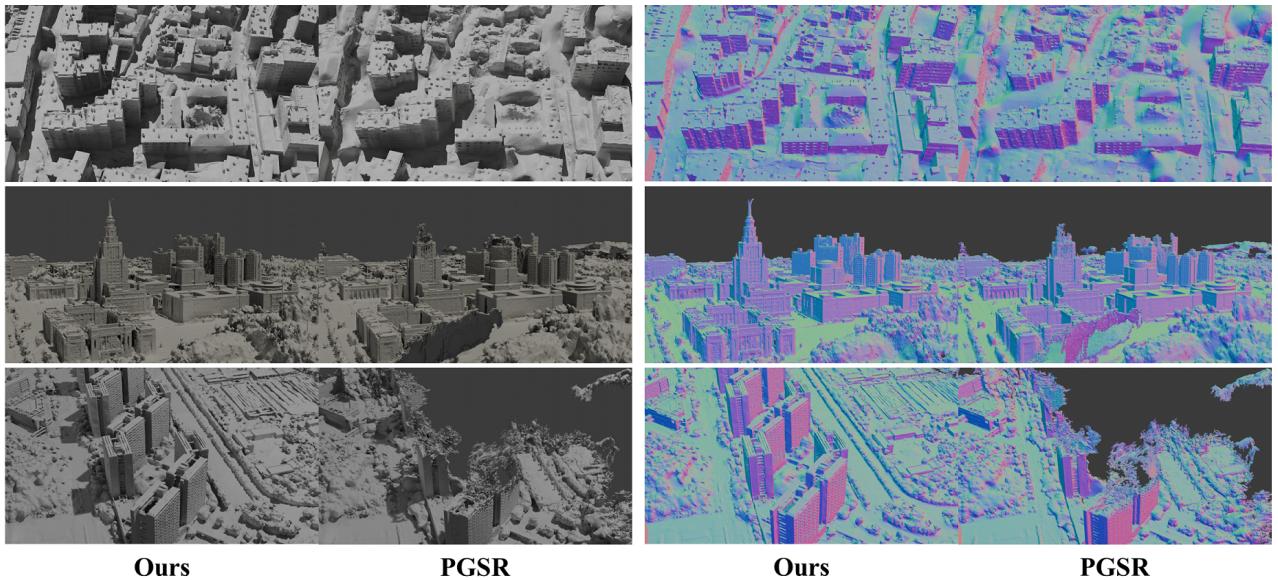Figure 16. Qualitative comparisons between ours and GOF [47].

Figure 17. Qualitative comparisons between ours and PGSR [3].

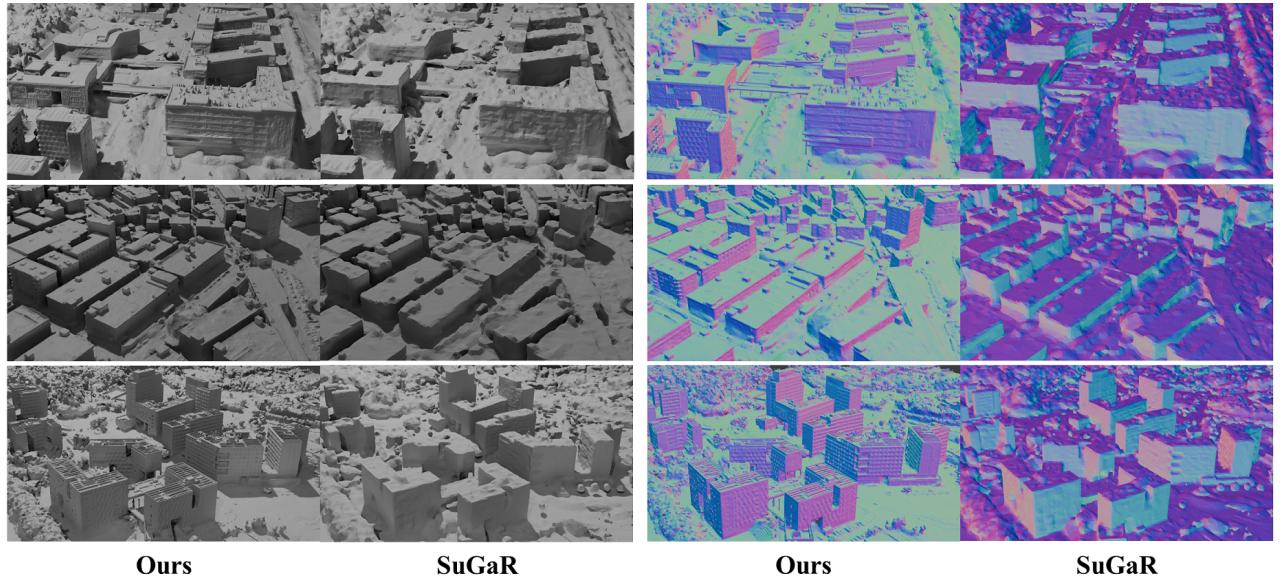**Ours**   **SuGaR**   **Ours**   **SuGaR**

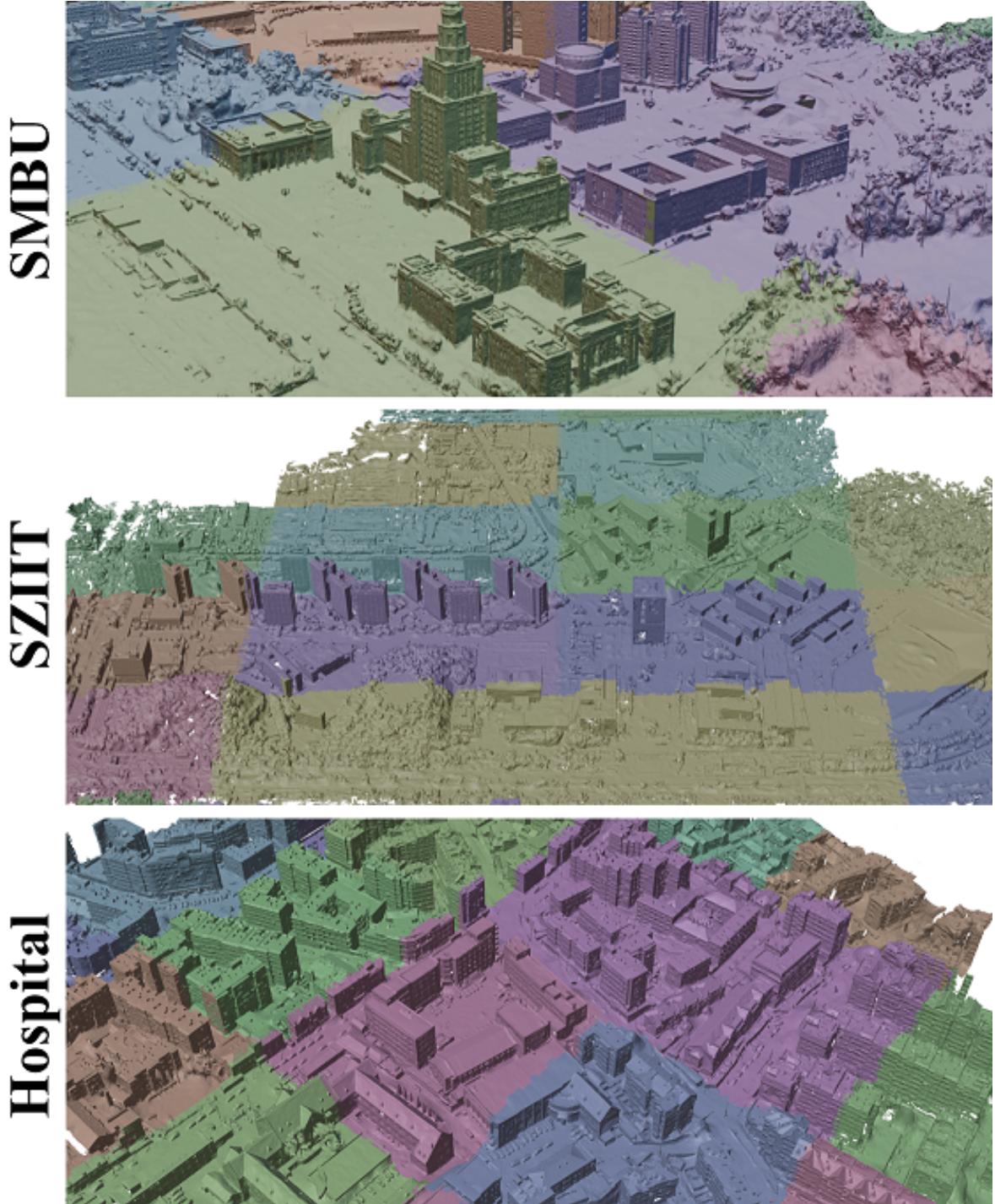Figure 18. Qualitative comparisons between ours and SuGaR [10].

Figure 19. **Example of partition mesh results.** Top to bottom: *SMBU*, *SZIIT* from GauU-Scene [38], and *Hospital* from UrbanScene 3D [21]