

Aerial-NeRF: Adaptive Spatial Partitioning and Sampling for Large-Scale Aerial Rendering

Xiaohan Zhang, Yukui Qiu, Zhenyu Sun, and Qi Liu, *IEEE Senior Member*

Abstract—Recent progress in large-scale scene rendering has yielded Neural Radiance Fields (NeRF)-based models with an impressive ability to synthesize scenes across small objects and indoor scenes. Nevertheless, extending this idea to large-scale aerial rendering poses two critical problems. Firstly, a single NeRF cannot render the entire scene with high-precision for complex large-scale aerial datasets since the sampling range along each view ray is insufficient to cover buildings adequately. Secondly, traditional NeRFs are infeasible to train on one GPU to enable interactive fly-throughs for modeling massive images. Instead, existing methods typically separate the whole scene into multiple regions and train a NeRF on each region, which are unaccustomed to different flight trajectories and difficult to achieve fast rendering. To that end, we propose Aerial-NeRF with three innovative modifications for jointly adapting NeRF in large-scale aerial rendering: (1) Designing an adaptive spatial partitioning and selection method based on drones' poses to adapt different flight trajectories; (2) Using similarity of poses instead of (expert) network for rendering speedup to determine which region a new viewpoint belongs to; (3) Developing an adaptive sampling approach for rendering performance improvement to cover the entire buildings at different heights. Extensive experiments have conducted to verify the effectiveness and efficiency of Aerial-NeRF, and new state-of-the-art results have been achieved on two public large-scale aerial datasets and presented SCUTic dataset. Note that our model allows us to perform rendering over 4 times as fast as compared to multiple competitors. Our dataset, code, and model are publicly available at <https://drliuqi.github.io/>.

Index Terms—View synthesis, large-scale scene rendering, neural radiance fields, fast rendering

I. INTRODUCTION

NEURAL Radiance Fields (NeRF) [1] synthesizes highly realistic 3D scenes from limited observations due to its implicit scene representation. NeRF has pervasive for rendering small objects and indoor scenes [2] [3] [4] [5] in illumination, reflections, and texture-less areas, which outperforms previous methods based on mesh [6] [7] [8] [9] and voxel [10] [11] [12] [13] with ever growing popularities [14] [15] [16] [17].

NeRF has also evolved from rendering small objects to more complex scenes. Such modeling can enable a variety of practical applications, including autonomous vehicle simulation [18] [19] [20], aerial surveying [21] [22], and embodied AI [23] [24]. Considering how to sample on objects at infinity, NeRF++ [25] proposes to compress unbounded scenes into bounded regions enhancing the rendering effect of distant

The authors are with the School of Future Technology, South China University of Technology, Guangzhou 511442, China. E-mail: {ftxiaoahnzhang, 202162311356, 202264690427}@mail.scut.edu.cn, drliuqi@scut.edu.cn (Corresponding author: Qi Liu).

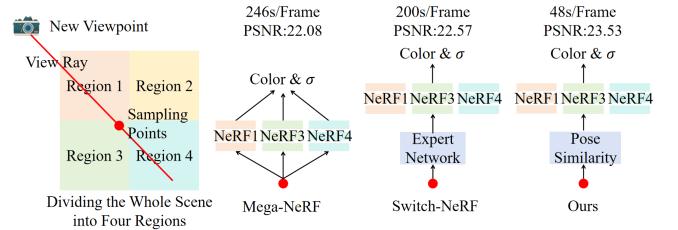


Fig. 1. Comparison of different methods for rendering new viewpoints. For a sampling point on the view ray, Mega-NeRF [28] uses NeRFs of all regions traversed by this ray to calculate the color and density of this sampling point, resulting in a plodding rendering speed. Switch-NeRF [29] applies an expert network to determine which region a sampling point belongs to and applies the corresponding NeRF to calculate its color and density, thereby improving the rendering speed. Our method creatively utilizes existing camera poses to match the region and the new viewpoint, which speeds up rendering. Moreover, ours is more robust for different aerial photography trajectories to achieve higher PSNR.

scenes. Mip-NeRF [2] replaces NeRF rays with view frustums and utilizes the structural information to achieve more accurate rendering results. This method can adaptively encode the inputs, via using low-frequency positional encoding in sparse sampling areas, and high-frequency positional encoding in dense sampling areas to achieve an anti-aliasing effect. NeRF-W [26] proposes appearance and transient embedding to handle changes in illumination and dynamic objects for the rendering quality enhancement of outdoor scenes. Block-NeRF [27] divides the street scene into multiple areas, and trains a NeRF separately in each area, enhancing the rendering accuracy of texture details. However, the deployment of NeRF-based models on large-scale aerial scenes is still impeded by two problems: (1) Using only a NeRF to render the whole scene can result in insufficient detail expression and excessive GPU memory consumption for high-resolution images in large-scale aerial datasets. (2) As for the far distance between the camera and the buildings, it becomes necessary to design adaptive sampling algorithm for NeRF to cover the buildings.

Motivated by the Block-NeRF [27], Mega-NeRF [28] evenly partitions the scene into multiple predefined regions to achieve arbitrarily large-scale scene rendering with only a single GPU, whereas it is unsuitable for uneven distributed drones. Switch-NeRF [29] performs region partitioning by learning to achieve good rendering results. Nevertheless, Switch-NeRF requires to input all images at once, which takes up high GPU memory consumption non-amicable to limited memory resources when the scene size grows. As shown in Fig. 1, both of them apply the (expert) network to determine the region which new perspective belongs to, which cost demanding inference time.

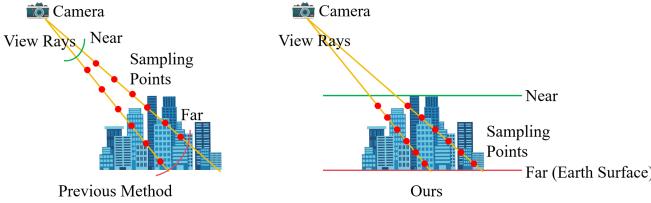


Fig. 2. Comparison between different sampling approaches. "Near" represents sampling origin along each view ray, and "Far" denotes the sampling end point. The previous sampling method sets the sampling range along each ray as a hyperparameter, resulting in a significant waste of sampling points in the air and cannot cover the entire buildings in the sampling range.

To address that, we design an efficient spatial partitioning and selection method, which clusters the poses of drones to partition the scene into multiple regions and selects cameras for each region based on boundary and similarity conditions. The purpose of selecting cameras is to ensure that enough cameras can observe a certain region to train the region's NeRF. To achieve sampling from unbounded space to bounded space, NeRF++ [25] is developed to partition the unbounded space into a bounded foreground and an unbounded background, where the foreground remains unaltered and the latter is compressed and transformed to a bounded space. However, the spatial gap between drones and the buildings results in numerous sampling points distributed in the air, leading to a waste of sampling points and the GPU memory increase. Moreover, the sampling range can not cover buildings adequately, as shown in Fig. 2. To that end, we propose an innovative strategy for ensuring that the sampling range covers all buildings and the sampling points are almost distributed on the buildings when cameras are at different heights. We sample between the highest building and the surface of the Earth, so that the sampling range covers all buildings and most of the sampling points are distributed on buildings.

Our work makes notable contributions summarized as follows:

- For large-scale aerial rendering, we propose an adaptive spatial partitioning and selection approach based on the camera's pose (position and orientation of the camera). The proposed method outperforms existing large-scale aerial rendering models by a large margin on the rendering speed, almost at 4 times. Besides, it is applicable to aerial datasets with diverse aerial photography trajectories. Additionally, under the appropriate number of divided regions, our method enables the rendering of arbitrarily large aerial scenes using a single GPU.

- We introduce a novel sampling strategy for aerial scenes, which enables to cover buildings by the sampling ranges from cameras at different heights. In a broader comparison against SOTA models, our approach is substantially more efficient (only 1/4 used sampling points and 2 GB GPU memory saving) and compares favourably in terms of multiple commonly-used metrics.

- We present SCUTic, a novel aerial dataset for large-scale university campus scenes, which includes 5.86 GB high-resolution oblique photography images. Unlike existing datasets, we collect data in a way that the camera trajectory is uneven, which can verify the robustness of rendering methods.

II. RELATED WORK

Aerial datasets differ from other image datasets because there is significant space between cameras and the buildings of interest. Performing perspective rendering on large-scale aerial datasets is challenging, and this work is gradually gaining attention. We propose a novel spatial division and selection approach, along with a novel sampling strategy, achieving state-of-the-art rendering results. Additionally, we create a new drone aerial dataset using a novel aerial capture strategy distinct from previous approaches. This dataset is utilized to validate the robustness of rendering models. We consider the most closely related works below.

A. NeRF for General Outdoor Scenes

NeRF [1] uses the MLP to map each sampling point's spatial position and view direction along view rays to color and density. By performing volume rendering integration along each view ray, the corresponding rendering image for a given camera view can be obtained. Due to the superiority of NeRF in view synthesis, many works improve its efficiency [30], accuracy [31] and apply it to 3D reconstruction tasks [32] [33].

NeRF also has evolved from rendering small objects to more complex scenes. We primarily focus on the application of NeRF in large-scale outdoor scenes. NeRF++ [25] analyzes the reasons for the success of NeRF and proposes a method to compress unbounded scenes into bounded regions. Mip-NeRF [2] proposes replacing NeRF rays with view frustums, utilizing the structural information to achieve more accurate rendering effects. NeRF-W [26] introduces algorithms to handle changes in illumination and dynamic objects, significantly enhancing the rendering quality of outdoor scenes. Block-NeRF [27] suggests dividing large scenes into many regions and training a separate NeRF for each region. This approach allows for rendering scenes of arbitrary size.

However, when rendering large-scale aerial scenes, the performance of these methods can not meet expectations. In typical outdoor scene datasets, buildings far from the camera are considered background and not the main focus of attention. In aerial datasets, buildings on the ground are far from the camera, but they are the scenes of interest for rendering. Therefore, sampling strategies and other related tactics need to be redesigned for aerial datasets.

B. NeRF for Large-Scale Aerial Scenes

This is a long-standing problem in computer vision [34] [35] [36] [37] [38] [39] [40]. For large-scale aerial scenes, it is essential to consider two key issues. The first issue is the sampling problem, determining how to ensure the sampling range covers objects on the ground. The second issue is region partitioning and selection because processing all images at once would require a high GPU memory. And the amount of data can exceed the expressive capacity of NeRF, resulting in blurry detail rendering.

Mega-NeRF [28] can render scenes of arbitrary size. This method uniformly divides the scene into several regions.

Then, it crops each photo to retain only the pixels visible in that particular region. The sampling strategy of Mega-NeRF [28] is similar to NeRF++ [25]. It involves sampling in both bounded regions and compressed unbounded regions. Eventually, the sampled points are fused to generate the color of the corresponding pixels. However, Mega-NeRF [28] sets the sampling range as a hyperparameter during sampling. This leads to the sampling range not covering objects and a significant number of sampled points in the air, causing an inaccurate rendering and a waste of sampling points. During rendering, Mega-NeRF [28] uses NeRFs of all regions that the view ray passes through to fuse the color and density of each sampling point on this ray, leading to a slow inference speed. And Mega-NeRF [28] requires a specific aerial photography trajectory, where the camera positions should be distributed as uniformly as possible in the space. If the distribution of drones in space is uneven, this algorithm fails.

Switch-NeRF [29] partitions the scene into several regions through a learning-based approach. This method requires inputting all the images and then learning the category of each pixel to partition the scene. As the scene size increases, this method requires computational resources to increase linearly. And the shape of the space divided by this method is unknown, adversely affecting the rendering results when the distribution of drones is uneven. Moreover, this method shares the same sampling strategy as Mega-NeRF, leading to the need for numerous sampling points and situations where the sampling range can not cover objects.

In response, we propose a novel method, Aerial-NeRF, with solid efficiency and robustness for large-scale aerial datasets. Firstly, our spatial partitioning and selection approach allows fast and accurate rendering. Secondly, our proposed sampling method enables sampling near objects, achieving high-quality rendering with minimal sample points. Moreover, our method can render aerial scenes of arbitrary size using only a single GPU. Finally, we introduce a new aerial dataset. The cameras' trajectory in our dataset differs from previous datasets, providing a good validation of the method's robustness.

III. METHOD

The pipeline of our method is shown in Fig. 3. We partition the space into multiple regions based on the distribution of drones and select corresponding cameras in each region based on boundary and similarity conditions. We also propose an adaptive sampling algorithm that allows the sampling range to cover the buildings and samples near the buildings.

A. Neural Radiance Field

We use the original NeRF [1] as our network architecture. NeRF models a scene using a consistent volumetric radiance field to capture the scene's geometry and appearance variations. During rendering, NeRF calculates a view ray for each pixel based on the camera pose, and performs sampling on this view ray. Then, the coordinates of sampling points and the direction of the view ray are passed through MLP to obtain the color $\mathbf{c}_i = (r, g, b)$ and density σ_i of these sampling

points. Finally, NeRF derives the pixel color $\hat{\mathbf{C}}(\mathbf{r})$ through the integration:

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=0}^{N-1} T_i(1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i \quad (1)$$

where δ_i is the distance between samples p_i and p_{i+1} . T_i represents the cumulative transparency of p_i , and

$$T_i = \exp\left(-\sum_{j=0}^{i-1} \sigma_j \delta_j\right) \quad (2)$$

NeRF employs a two-stage hierarchical sampling procedure to sample on view rays. In a coarse stage, uniform sampling is performed within the sampling range, while in a fine stage, inverse distribution sampling is performed based on the density of the sampling points from the coarse stage. During the training process, the model is minimized by the loss function L_{mse} with the ground truth $C(\mathbf{r})$:

$$L_{mse} = \sum_{\mathbf{r} \in R} \|C(\mathbf{r}) - \hat{\mathbf{C}}(\mathbf{r})\| \quad (3)$$

where R represents the batches of pixels. However, NeRF computes the color for each pixel independently, which leads to a loss of structural information in the images. We incorporate the $S3IM$ loss function [41] L_{S3IM} to constrain the structural information:

$$L_{S3IM} = 1 - \frac{1}{M} \sum_{m=1}^M SSIM(\mathbb{P}^{(m)}(\hat{\mathbf{C}}(\mathbf{r})), \mathbb{P}^{(m)}(C(\mathbf{r}))) \quad (4)$$

The final loss function L is defined as:

$$L = \lambda_{MSE} L_{MSE} + \lambda_{S3IM} L_{S3IM} \quad (5)$$

where λ_{MSE} , λ_{S3IM} are the weights of L_{MSE} , L_{S3IM} .

B. Spatial Partitioning and Selection

We divide the scene into multiple regions based on the distribution of drones, and then perform clustering on the XY-plane using k-means clustering method [42]. The scene is divided into N regions $cluster_i, i = 1, 2, \dots, N$, and N region centroid points $\mathbf{o}_i, i = 1, 2, \dots, N$ (\mathbf{o}_i is the coordinate on the XY-plane) are obtained. There may be cameras from $cluster_j$ that can see $cluster_i$. We aim to train the i -th region's NeRF using enough cameras which can observe a certain region. Therefore, we need to observe the boundary cameras between regions. The characteristic of boundary cameras is that they are close to other regions and may be able to view other regions' scenes. For $cluster_j$, if there exists a camera whose coordinate is denoted as \mathbf{p}_j that satisfies $\|\mathbf{p}_j - \mathbf{p}_i\| < \alpha$ with a camera \mathbf{p}_i in $cluster_i$ and a threshold α , then the camera \mathbf{p}_j is considered as a boundary camera between $cluster_i$ and $cluster_j$.

Next, we select the boundary cameras that can observe $cluster_i$. If the boundary camera \mathbf{p}_j is in $cluster_j$, it needs to be determined whether this camera can view $cluster_i$. The orientation of this camera in its coordinate system is $\mathbf{d}_c = (0, 0, 1)$. We denote the rotation and translation from

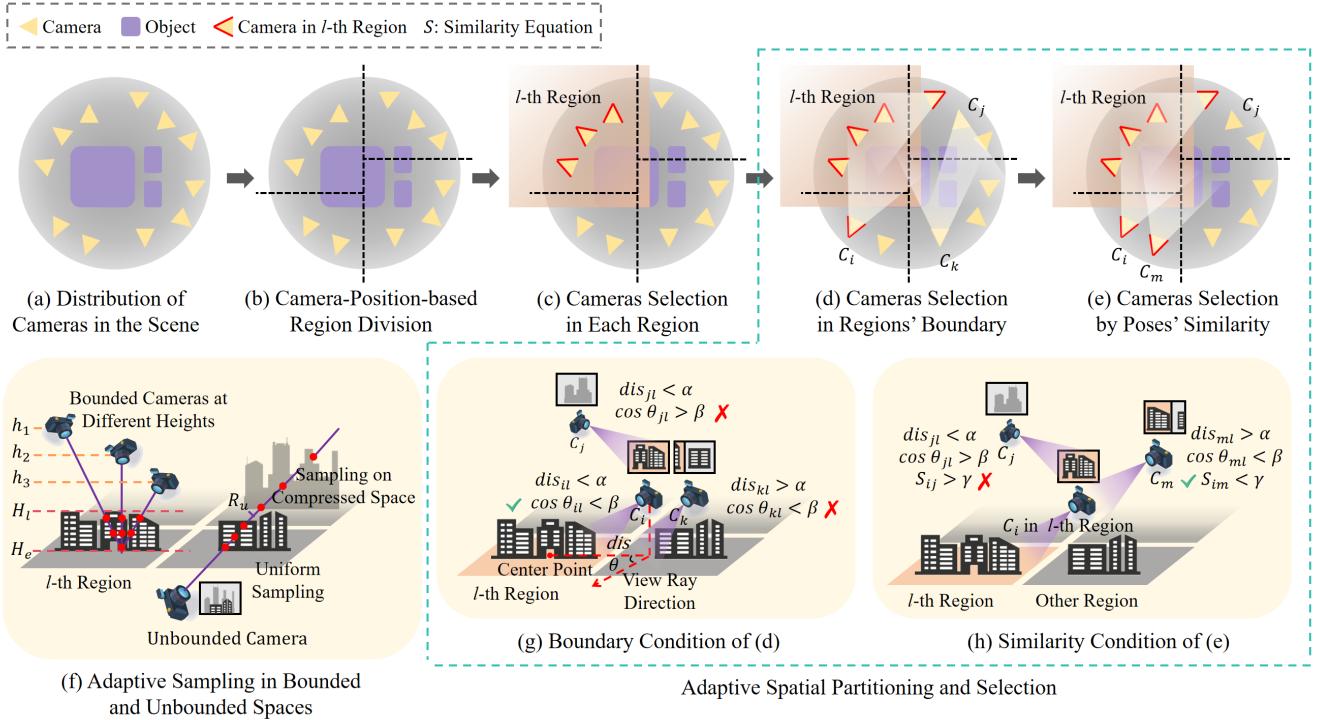


Fig. 3. The pipeline of our method. We propose an adaptive spacial partitioning and selection method that makes our method applicable to aerial datasets of different trajectories. (a) (b) divide the entire scene into multiple areas based on the poses of drones. Next, we select cameras that can observe l -th region, and use these cameras to train the NeRF of l -th region. (c) selects the cameras in l -th region. (d) selects the boundary cameras that can see the l -th region. (e) utilizes the boundary cameras to select more cameras that can view the l -th region. (f) samples between H_1 and H_2 in bounded space, and samples on buildings to infinity in unbounded space. (g) is the condition for determining whether the boundary cameras belongs to the l -th region. When the distance dis and angle θ are within the threshold, it indicates that this boundary camera can observe the l -th region. (h) is to determine whether the non-boundary cameras outside the l -th region belongs to this region. We use the similarity equation to find cameras in other regions that are similar to the boundary camera of the l -th region.

the camera coordinate system to the world coordinate system as \mathbf{R} and \mathbf{t} , respectively. Then, the orientation of the camera in the world coordinate system \mathbf{d}_w is:

$$\mathbf{d}_w = \mathbf{R}\mathbf{d}_c + \mathbf{t} \quad (6)$$

We denote the vector pointing from camera \mathbf{p}_j to \mathbf{o}_i as \mathbf{a} . If the angle θ between \mathbf{d}_w and \mathbf{a} satisfies $\cos \theta > 0$, it indicates that camera \mathbf{p}_j can observe region $cluster_i$ and then add this camera to $cluster_i$. Since this camera can also view the scenes from $cluster_j$, it belongs to both $cluster_i$ and $cluster_j$.

Due to the limited number of boundary cameras, there exist some cameras in $cluster_j$ that, although are not boundary cameras, can view $cluster_i$. These cameras need to undergo region redefinition as well. For two cameras \mathbf{p}_1 and \mathbf{p}_2 , we use the rotation \mathbf{R} , translation \mathbf{t} , and capture time $time$ to measure their similarity. \mathbf{R} and \mathbf{t} help to keep the similarity of the cameras' spatial position and orientation. $time$ ensures that the shooting times are not too far apart and maintain small differences in light changes between cameras. The similarity error S between camera \mathbf{p}_1 and camera \mathbf{p}_2 is calculated as:

$$S = \left\| \begin{bmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ 0 & time_1 \end{bmatrix} - \begin{bmatrix} \mathbf{R}_2 & \mathbf{t}_2 \\ 0 & time_2 \end{bmatrix} \right\| \quad (7)$$

The smaller the value of S , the more similar the two cameras are. For each boundary camera, we calculate n_p cameras with

the most minor similarity error. Then, they are assigned to the same region as the corresponding boundary cameras. On the basis of that, we divide the entire large-scale scene into multiple regions and train a separate NeRF on each region. The advantage of dividing space based on drones' pose is that our model can be applied to any aerial photography trajectory and the computational resources do not increase with the scene's size.

Spatial Selection for New Viewpoints. During training, we divide the cameras into corresponding regions and train the NeRF for each region. During inference, For a new viewpoint, we creatively design an accurate and fast spatial selection method based on the known cameras in the space to determine which region it belongs to and then render this viewpoint with the corresponding NeRF, as shown in Fig. 4. Firstly, we should determine n_s cameras most similar to the new viewpoint at each region by computing the similarity error S in equation (7), and take the mean of n_s cameras as the region similarity error \bar{S} between the new viewpoint and each region. If the \bar{S} of a particular region satisfies $\bar{S} < \gamma$, where γ is the threshold, It indicates that the new viewpoint can observe the scene of this region. Therefore, the new viewpoint is rendered by this region's NeRF. When the above condition is satisfied for several regions, we simply render the new viewpoint with the NeRFs of these regions and then take the average of all rendering results to get the final rendering result.

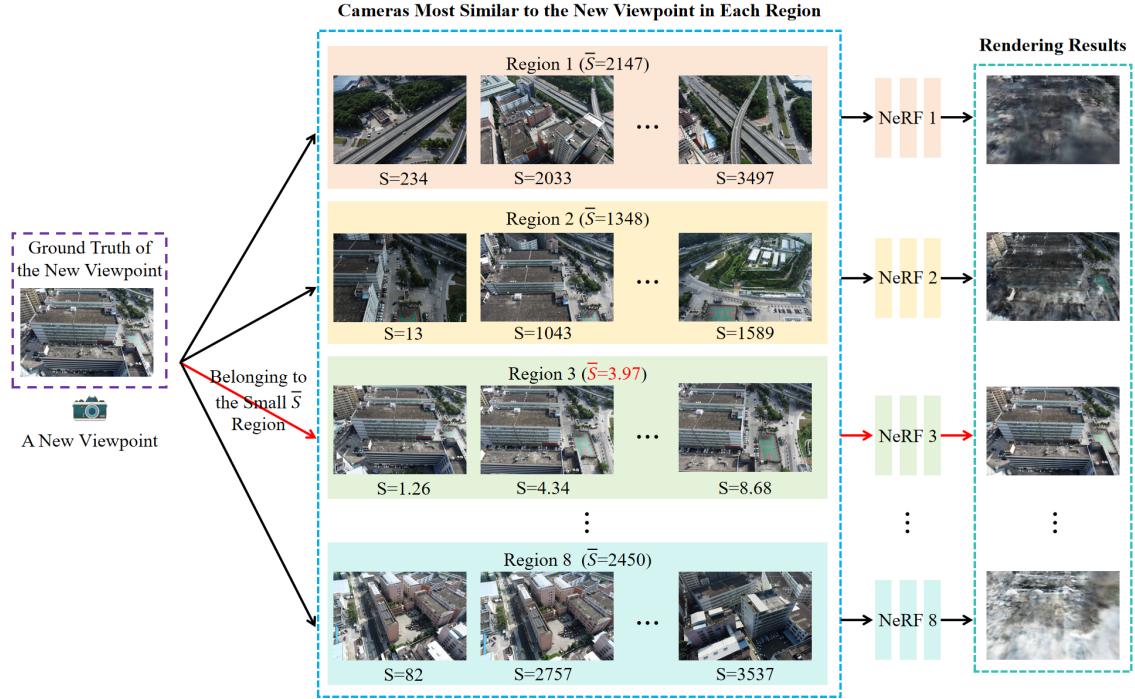


Fig. 4. Visualization of our spatial selection strategy. The algorithm's input is the pose of a new viewpoint, and the output is the rendering image of this viewpoint. In each region, we find the $n_s = 5$ cameras with the smallest S (calculated by (7)) relative to this viewpoint. The smaller the S , the higher the similarity between cameras, and the more common view areas there are. For instance, the scene viewed from the new viewpoint is almost identical to that captured by a camera with $S = 1.26$. To avoid randomness, taking the average of $n_s = 5$ cameras' S as the region similarity error \bar{S} between this viewpoint and each region. When \bar{S} is small, this camera belongs to this region, and the NeRF of this region is used to render this viewpoint. As can be seen from the images, the smaller the \bar{S} , the smaller the difference between images of the new viewpoint and this region's cameras, indicating that the new viewpoint belongs to this space.

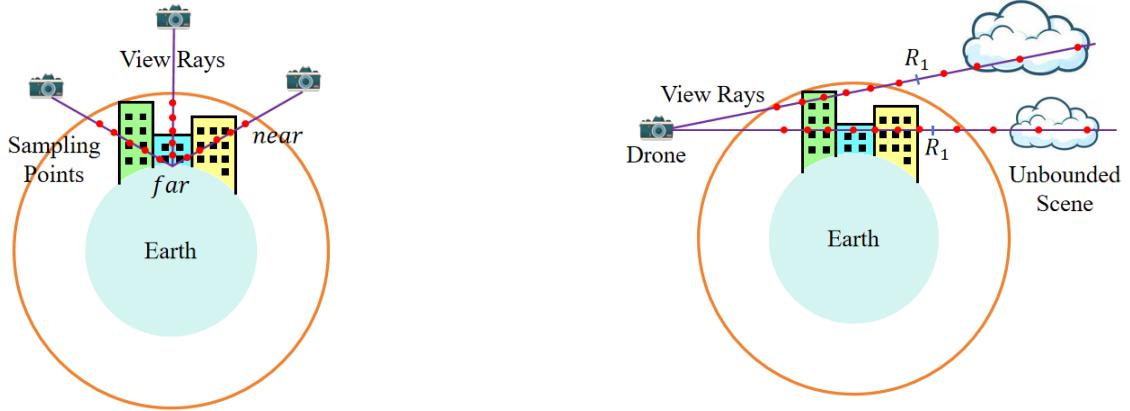


Fig. 5. Sampling strategy on bounded regions. The intersection of the drone's ray with the outer sphere *near* is the starting point for sampling, and the intersection with the Earth *far* is the ending point for sampling. Each drone's ray is sampled to cover buildings on the ground.

C. Adaptive Sampling

During the process of training a NeRF for a region, sampling is performed along each ray for every pixels. The previous sampling methods do not cover the object adequately, resulting in wasted sampling points and loss of accuracy. We propose an adaptive sampling method to ensure cameras at different heights sampling near objects.

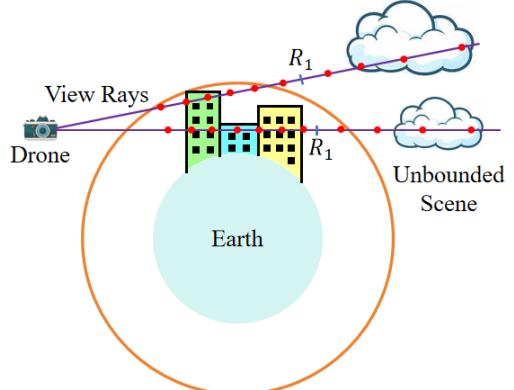


Fig. 6. Sampling strategy for unbounded regions. When the view rays do not intersect with the Earth or are directed towards the sky, objects in unbounded regions, such as clouds, are visible. We uniformly sample from the camera origin to R_1 . The sampling interval gradually increases in the $[R_1, +\infty]$.

Sampling on Bounded Space. As shown in Fig. 5, we assume the Earth is a sphere with a radius of R_{Earth} . The height h of the tallest building in the current region can be obtained from the sparse point cloud calculated by COLMAP [43]. A new sphere with the center at the center of the Earth and a radius of $R = R_{Earth} + h$ is constructed. A camera's position in space is given by (x_0, y_0, z_0) , and its view ray direction is represented by (a, b, c) . The parameter equation

of this view ray with respect to t can be expressed as:

$$\begin{cases} x = x_0 + at \\ y = y_0 + bt \\ z = z_0 + ct \end{cases} \quad (8)$$

The center coordinate of the Earth is (r_1, r_2, r_3) , then the Earth can be built as:

$$(x - r_1)^2 + (y - r_2)^2 + (z - r_3)^2 = R_{Earth}^2 \quad (9)$$

The outer sphere can be represented as:

$$(x - r_1)^2 + (y - r_2)^2 + (z - r_3)^2 = R^2 \quad (10)$$

The intersection of the view ray with the outer sphere is the nearest sampling point "near", and the intersection with the Earth is the farthest sampling point "far". Substituting equation (8) into equations (9) and (10), *near* and *far* can be solved by

$$near = \frac{-B - \sqrt{B^2 - 4AC_{Earth}}}{2A} \quad (11)$$

$$far = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \quad (12)$$

where

$$A = a^2 + b^2 + c^2$$

$$B = 2(a(x_0 - r_1) + b(y_0 - r_2) + c(z_0 - r_3))$$

$$C_{Earth} = (x_0 - r_1)^2 + (y_0 - r_2)^2 + (z_0 - r_3)^2 - R_{Earth}^2$$

$$C = (x_0 - r_1)^2 + (y_0 - r_2)^2 + (z_0 - r_3)^2 - R^2$$

Thus, the sampling range along the view ray is $[near, far]$. Similar to NeRF [1], we first uniformly sample within this range, then calculate the density of the sampled points, and perform inverse distribution sampling based on the density.

Sampling on Unbounded Space. When the camera observes unbounded regions such as the sky and clouds, the view ray does not intersect with the Earth. Therefore, we design a sampling method for unbounded regions, as shown in Fig. 6. We divide the view ray into foreground $[0, R_1]$ and background $[R_1, +\infty]$. We uniformly sample in the foreground, and sample points gradually further away on the background, extending to unbounded space. To that end, we compress the space as:

$$cam(t) = \begin{cases} t, & 0 \leq t \leq R_1 \\ R_1 + \frac{1}{R_1} - \frac{1}{t}, & \text{otherwise} \end{cases} \quad (13)$$

where $cam(t)$ represents the compressed space. When $t > R_1$, we sample uniformly in the compressed space. Let the sampling point be s , $s \in (R_1, R_1 + \frac{1}{R_1})$, then the corresponding sampling point mapped back to the original space is:

$$t = \frac{1}{-s + R_1 + \frac{1}{R_1}} \quad (14)$$

Hence, the sampling range for t is $(R_1, +\infty)$, and the space between sampling points becomes increasingly larger.

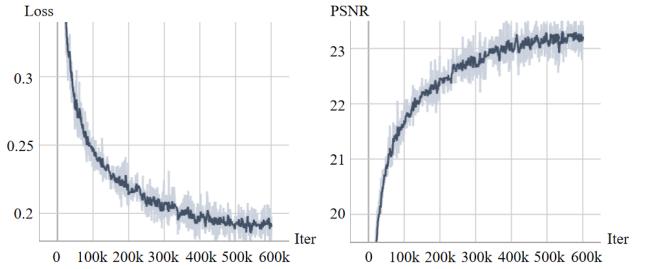


Fig. 7. The relationship between the number of iterations, loss function, and PSNR. The model converges after 600k iterations.

IV. EXPERIMENTS

A. Datasets

56 Leonard [44] consists of aerial datasets taken at different altitudes. When collecting data, the strategy is to move the camera in a circular motion and gradually elevate the camera from a low altitude to a high altitude. The range of cameras in this dataset is $300m \times 300m$.

Residence from UrbanScene3D [45] consists of large-scale scene pictures captured at the same altitude with a uniform camera trajectory. The range of cameras in this dataset is $250m \times 400m$.

SCUTic is captured at the same altitude utilizing uneven aerial trajectories. We conduct aerial photography around each building. In areas where buildings are dense, the density of cameras is high, while in areas where buildings are sparse, the density of cameras is low. This dataset contains 5.86 GB high-resolution images taken from the South China University of Technology International Campus by a DJI Mini 2 drone with $500m \times 600m$ camera range. The uneven cameras' distribution and the large camera range pose challenges for view rendering and 3D reconstruction to test the robustness and generalizability of different methods.

B. Metrics and Settings

Metrics. Our results on novel view synthesis are quantitatively evaluated by PSNR [46], SSIM [47], and the LPIPS implementation of VGG [48]. PSNR is utilized to calculate the mean squared error between two images in logarithmic space. SSIM is more concerned with structural similarity. LPIPS is used to assess perceptual similarity.

Settings. The scene is divided into several regions and a separate NeRF is trained on each region. For the Residence dataset, the scene is partitioned into 8 regions. For the 56 Leonard and SCUTic datasets, the scene is divided into 4 regions. Similar to NeRF [1], the 8-layer MLP is utilized for feature extraction, where each layer produces features with 256 channels. The 48-dimensional appearance encoding is employed to adapt the model to different lighting conditions. We train 600k iterations for each dataset with 4096 batch size. The loss and PSNR versus the number of iterations are shown in Fig. 7, respectively. The Adam is applied as optimizer [49] with a learning rate decaying exponentially from 5×10^{-4} to 5×10^{-5} . The number of sampling points at the coarse sampling stage is 64, and at the fine stage is 128. Our model is trained by a single RTX 3090.

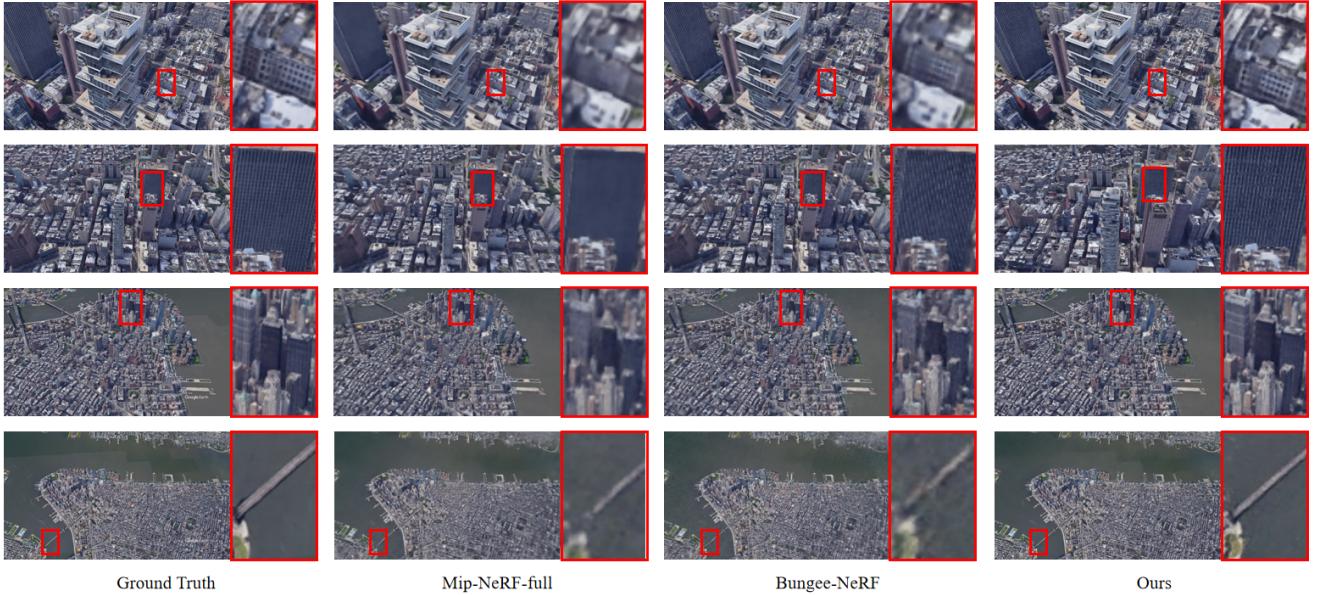
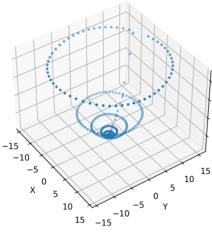
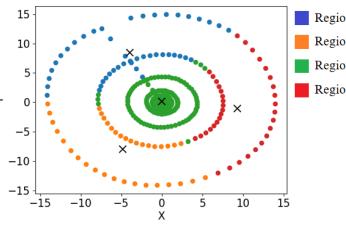


Fig. 8. The comparison of rendering results on four different altitudes. Our rendering results show clearer texture details at lower altitude and more complete image information at higher altitude.

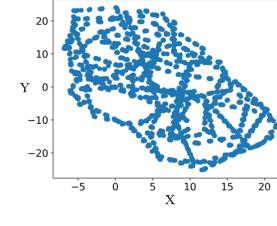
The Distribution of all Cameras



Partitioning into Four Regions



The Distribution of all Cameras



Partitioning into Eight Regions

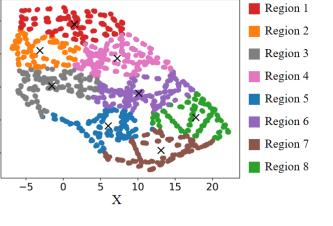


Fig. 9. Our spatial partitioning on the 56 Leonard dataset. The left figure represents the distribution of drones in this dataset and the right figure shows the cameras we selected in each region. X, Y and Z axes represent the range of drone distribution, with a unit of 10 meters. Those cameras at low altitudes are relatively close and are grouped into one region. At high altitudes, cameras at different heights are grouped into one region.

TABLE I
COMPARISON OF DIFFERENT METHODS ON THE 56 LEONARD DATASET.
THE BOLD DATA IN EACH COLUMN REPRESENTS THE BEST FOR EACH METRIC.

Method	PSNR↑	SSIM↑	LPIPS↓
NeRF [1] (D=8, Skip=4)	21.702	0.320	0.636
Mip-NeRF-small [2]	23.337	0.709	0.354
Mip-NeRF-large [2]	23.507	0.718	0.346
Mip-NeRF-full [2]	23.665	0.732	0.328
Bungee-NeRF [44]	24.513	0.815	0.160
Ours	25.333	0.832	0.148

C. Results

Results on 56 Leonard. The comparison of rendering results at four different altitudes is shown in Fig. 8. Our method achieves rendering the texture of buildings more clearly in low-altitude areas and structural information more completely in high-altitude areas, respectively. Bungee-NeRF [44] classifies cameras at each altitude into one category, and uses a coarse-to-fine method to train NeRFs from high-

Fig. 10. Our spatial partitioning on the Residence dataset. The left figure represents the distribution of drones on the XY-plane, while the right figure illustrates the clustering of cameras. The X and Y axes represent the range of drone distribution, with a unit of 10 meters. We divide the cameras relatively evenly into the corresponding space.

altitude areas to low-altitude areas sequentially. However, this space partitioning method based on altitude can result in a significant loss of texture information. The reason is that the texture details decrease as the altitude increases. Bungee-NeRF [44] only utilizes images from one altitude to train the corresponding NeRFs, resulting in less information to describe each building, and blurry and incomplete rendering results. Our adaptive spatial method effectively solves this problem by grouping cameras at different altitudes into one category, as shown in Fig. 9. At the lowest altitude, the cameras are densely distributed, and there is a lot of shared information between adjacent cameras. Grouping them into one category can yield accurate rendering results. As the altitude increases, the range of the scene expands while the texture details of the buildings decrease. In this case, grouping cameras at different heights into one category provides both global structure information and texture details, thereby improving the quality of rendering.

From Table I, as compared to NeRF [1] and Mip-NeRF [2], our method achieves 1.668, 0.1, and 0.18 dB increase in PSNR, SSIM, and LPIPS, respectively. Compared to Bungee-

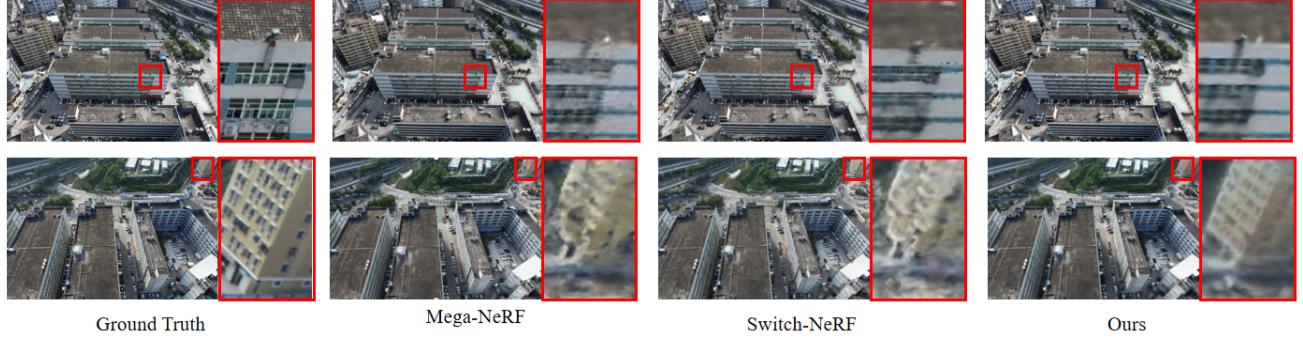


Fig. 11. Comparison of results on the Residence dataset. Our designed adaptive sampling and spatial partitioning strategy can improve rendering accuracy.

TABLE II
COMPARISON OF DIFFERENT METHODS ON THE RESIDENCE DATASET. EXTENSIBILITY INDICATES WHETHER A SINGLE GPU CAN RENDER SCENES OF ANY SIZE. THE BOLD DATA IN EACH COLUMN REPRESENTS THE BEST FOR EACH METRIC.

Method	PSNR↑	SSIM↑	LPIPS↓	GPU Memory↓	Rendering Time↓	Extensibility
NeRF [1]	19.01	0.593	0.488	26G	41s	✗
NeRF++ [25]	18.99	0.586	0.493	29G	44s	✗
Mega-NeRF [28]	22.08	0.628	0.489	10G	246s	✓
Switch-NeRF [29]	22.57	0.654	0.457	32G	200s	✗
Ours	23.52	0.742	0.261	8G	48s	✓

NeRF [44] with 120 hours for training, ours achieves 0.82, 0.017, and 0.012 dB increase in PSNR, SSIM, and LPIPS, respectively, and only requires 32 training hours which indicates the effectiveness of our adaptive spatial partitioning algorithm.

Results on Residence. Fig. 10 shows the results of our spatial partitioning on the Residence dataset, where the pose distribution of drones is relatively uniform. In this case, various models can achieve high-quality rendering. Fig. 11 compares the rendering results of our method with other approaches. The previous methods do not render the texture details finely enough and result in incomplete rendering of distant buildings. The reason is that these methods set the sampling range as a hyperparameter and the sampling points cannot cover the entire scene, affecting the quality of rendering. We design the adaptive sampling method which ensures that the sampling range at different altitudes covers buildings to deal with this issue.

In Table II, as compared to the SOTA Switch-NeRF [29], our method uses 1/4 sampling points to save 24 GB GPU memory but improves 0.95, 0.088, and 0.196 dB in PSNR, SSIM, and LPIPS, respectively. In addition, we compare the rendering time of different methods. Since NeRF [1] and NeRF++ [25] do not perform spatial partitioning, there is no need to select regions for new perspectives, and they achieve fast rendering. Mega-NeRF [28] applies the NeRFs on regions where a ray passes through to jointly calculate the color of a pixel. This results in a linear increase in rendering time as the number of regions increase. Switch-NeRF [29] requires an expert network to determine which region each sampling point belongs to, and then calculates the color and density of the sampling point with the corresponding NeRF. The large expert network slows down the rendering speed. Our method creatively utilizes existing cameras' poses to determine which

region a new viewpoint belongs to, which does not rely on a network and dramatically speeds up the rendering. Compared with Mega-NeRF [28] and Switch-NeRF [29], our method increases the rendering speed by 4 times. We also compare the extensibility of different methods if they can enable to render arbitrarily large scenes using a single GPU. NeRF [1], NeRF++ [25] and Switch-NeRF [29] train all images at once and the GPU memory increases as the scene size increases. Mega-NeRF [28] first divides the whole scene into several regions and trains the corresponding NeRFs using each region's images, which significantly saves the GPU memory. That is to say, as long as the spatial division method is appropriate, any large aerial scene can be rendered based on a single GPU. However, Mega-NeRF [28] divides the space evenly into multiple parts. The rendering accuracy is significantly lower down when the distribution of the drones' trajectory is uneven. Our spatial partitioning and selection method based on drone pose distribution answers in the affirmative to address that, and is robust to the cameras' trajectory of aerial datasets.

Results on SCUTic. Existing aerial datasets have a relatively uniform distribution of cameras in space, and various methods can achieve satisfactory rendering results. To verify the robustness of those methods, we create a new dataset, named SCUTic, in a way that the camera distribution is uneven. The rendering results on this dataset are shown in Fig. 12. It can be seen that the SOTA method Switch-NeRF [29] is easily influenced by the aerial flight trajectory. The reason is that the uneven aerial flight trajectory results in a large difference in information across different regions. Switch-NeRF [29] needs to integrate the uneven information from those regions when view rays pass through to synthesize a new viewpoint. We train a separate NeRF for each area to learn different spatial information. Our spacial selection algorithm based on pose similarity can adaptively assign the

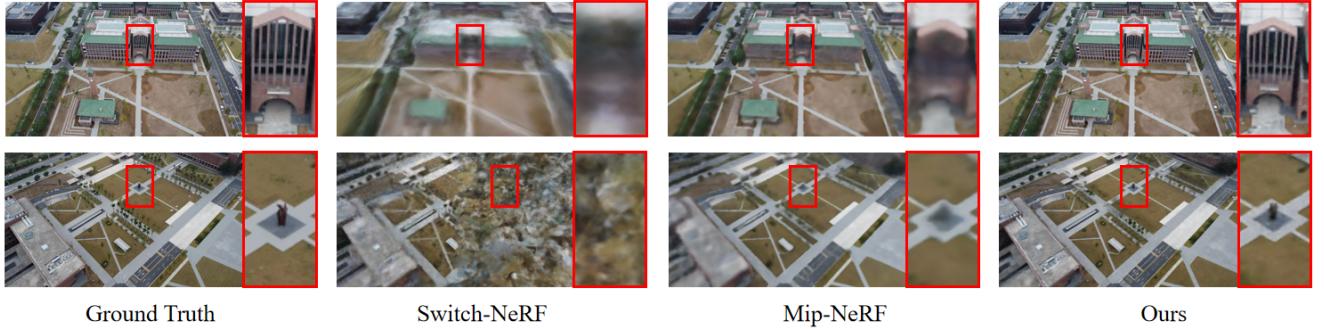


Fig. 12. Comparison of results on the SCUTic dataset. When the distribution density of drones in space is uneven, our spatial partitioning method based on camera poses is more robust compared to previous spatial partitioning methods.

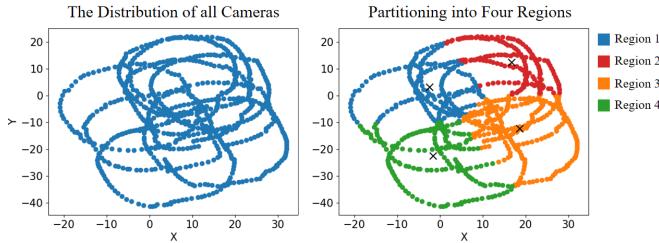


Fig. 13. The distribution of drones and region division of our method on the SCUTic dataset. This dataset is obtained by drones circling around each building. When the buildings are dense, the camera distribution is dense. When the buildings are sparse, the camera distribution is sparse. We evenly divide the cameras into their respective regions based on our spatial partitioning method.

TABLE III
COMPARISON RESULTS ON THE SCUTIC DATASET. PARTITIONING REPRESENTS SPACIAL PARTITIONING AND SELECTION METHODS. THE BOLD DATA IN EACH COLUMN REPRESENTS THE BEST FOR EACH METRIC.

Method	Partitioning	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF [2]	-	22.40	0.717	0.318
Mega-NeRF [28]	Uniform	-	-	-
Switch-NeRF [29]	Learning	18.8	0.479	0.598
Ours	Pose	27.62	0.847	0.108

new viewpoint to the corresponding region and use the NeRF of this region for rendering, thereby solving the problem of information difference. The spatial partitioning of our method is shown in Fig. 13. For example, in Region 2, the cameras are densely distributed and the amount of information is large, therefore the number of cameras assigned to this region is relatively fewer. In contrast, in Region 4, the cameras are sparse and the amount of information is small, therefore more cameras are allocated to increase the information in this region. The rendering performance of each area is improved by reasonably allocating cameras.

In Table III, Mega-NeRF [28] partitions the space evenly, while Switch-NeRF [29] divides the space through learning. These two approaches are greatly influenced by the flight trajectory and Mega-NeRF [28] even fails to render. Our method partitions the space based on the poses of drones to satisfy different flight trajectories. Compared to Switch-NeRF

TABLE IV
COMPARISON OF RENDERING RESULTS WITH DIFFERENT NUMBERS OF REGION PARTITIONS. THE BOLD TEXT IN EACH COLUMN REPRESENTS THE BEST DATA FOR EACH METRIC.

Method	PSNR↑	SSIM↑	LPIPS↓
Ours-4	22.62	0.722	0.293
Ours-8	23.52	0.742	0.261
Ours-16	23.66	0.763	0.229

TABLE V
ABLATION EXPERIMENTS FOR EACH MODULE OF OUR METHOD. THE BOLD DATA IN EACH COLUMN REPRESENTS THE BEST FOR EACH METRIC.

Method	PSNR↑	SSIM↑	LPIPS↓
w/o Spacial Partitioning	22.21	0.718	0.326
w/o Spacial Selection	22.97	0.728	0.287
w/o Adaptive Sampling	21.36	0.647	0.357
Full Method	23.52	0.742	0.261

[29], our method achieves 5.22, 0.368, and 0.49 dB increase in PSNR, SSIM, and LPIPS, respectively. The rendering results of our method on all datasets are shown in Fig. 14.

V. ABLATION STUDIES

Our method partitions the scene based on the distribution of drones and trains a separate NeRF on each area. As shown in Table IV, the more the number of divided regions is, the more texture details of each region's NeRF can perceive, and the higher the rendering accuracy is achieved.

Table V shows the effect of each module in the proposed pipeline. They are:

Spacial Partitioning. Our method divides the space based on the pose of cameras, which can reasonably distribute the information of the scene to each region and improve the rendering effect. Compared with evenly dividing the scene into several areas, our method achieves 1.31, 0.024, 0.065 dB increase in PSNR, SSIM, and LPIPS, respectively.

Spacial Selection. We design the boundary condition and similarity condition to select cameras for a certain region. Compared with only considering camera coordinates to allocate cameras, our method achieves 0.55, 0.014, 0.026 dB increase in PSNR, SSIM and LPIPS, respectively.



Fig. 14. Rendering results of our method on three datasets. Our method can achieve vivid rendering results at the same height, different heights, and in situations where the drone distribution is uneven.

Adaptive Sampling. Our method can make the sampling range cover the entire buildings and distribute numerous sampling points on the buildings, further improving the rendering quality. Compared with setting the sampling range as a hyperparameter, our method achieves 2.16, 0.095, 0.096 dB increase in PSNR, SSIM and LPIPS, respectively.

VI. CONCLUSION

In this paper, we propose an efficient and robust rendering method, termed Aerial-NeRF, for processing large-scale aerial datasets. Herein, our method using an adaptive spatial partitioning and selection approach outperforms existing large-scale aerial rendering competitors by a large margin on the rendering speed, almost at 4 times. Additionally, under the appropriate number of divided regions, Aerial-NeRF enjoys the rendering of arbitrarily large scenes using a single GPU. Meanwhile, we introduce a novel sampling strategy for aerial scenes, which enables to cover buildings by the sampling ranges from cameras at different heights. In a broader comparison against SOTA models, our approach is substantially more efficient (only 1/4 used sampling points and 2 GB GPU memory saving) and compares favourably in terms of multiple commonly-used metrics. Finally, we present SCUTic, a novel aerial dataset for large-scale university campus scenes with uneven camera trajectory, which can verify the robustness of rendering methods.

REFERENCES

- [1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” *Commun. ACM*, vol. 65, no. 1, pp. 99–106, 2021.
- [2] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, “Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5855–5864.
- [3] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *ACM Trans. Graph.*, vol. 41, no. 4, pp. 1–15, 2022.
- [4] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, “TensorF: Tensorial radiance fields,” in *Proc. Eur. Conf. Comput. Vis.* Springer, 2022, pp. 333–350.
- [5] G.-W. Yang, W.-Y. Zhou, H.-Y. Peng, D. Liang, T.-J. Mu, and S.-M. Hu, “Recursive-NeRF: An efficient and dynamically growing NeRF,” *IEEE Trans. Vis. Comput. Graph.*, 2022.
- [6] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, “Mitsuba 2: A retargetable forward and inverse renderer,” *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–17, 2019.
- [7] S. Liu, T. Li, W. Chen, and H. Li, “Soft rasterizer: A differentiable renderer for image-based 3d reasoning,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7708–7717.
- [8] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, “Differentiable monte carlo ray tracing through edge sampling,” *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–11, 2018.
- [9] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, “Unstructured lumigraph rendering,” in *Proc. 28th Annu. Conf. Comput. Graph. Interactive Techn.*, 2023, pp. 497–504.
- [10] S. M. Seitz and C. R. Dyer, “Photorealistic scene reconstruction by voxel coloring,” *Int. J. Comput. Vis.*, vol. 35, pp. 151–173, 1999.
- [11] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, “Neural volumes: Learning dynamic renderable volumes from images,” *ACM Trans. Graph.*, 2019.
- [12] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer, “Deepvoxels: Learning persistent 3d feature embeddings,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2437–2446.
- [13] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines,” *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–14, 2019.
- [14] N. Deng, Z. He, J. Ye, B. Duinkharjav, P. Chakravarthula, X. Yang, and Q. Sun, “Fov-NeRF: Foveated neural radiance fields for virtual reality,” *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 11, pp. 3854–3864, 2022.
- [15] H. Zhong, J. Zhang, and J. Liao, “VQ-NeRF: Neural reflectance decomposition and editing with vector quantization,” *IEEE Trans. Vis. Comput. Graph.*, 2023.
- [16] J. Zhang, X. Li, Z. Wan, C. Wang, and J. Liao, “Text2NeRF: Text-driven 3d scene generation with neural radiance fields,” *IEEE Trans. Vis. Comput. Graph.*, 2024.
- [17] K. Wang, S. Peng, X. Zhou, J. Yang, and G. Zhang, “NeRFCap: Human performance capture with dynamic neural radiance fields,” *IEEE Trans. Vis. Comput. Graph.*, 2022.
- [18] W. Li, C. Pan, R. Zhang, J. Ren, Y. Ma, J. Fang, F. Yan, Q. Geng, X. Huang, H. Gong *et al.*, “AADS: Augmented autonomous driving simulation using data-driven algorithms,” *Science robotics*, vol. 4, no. 28, p. eaaw0863, 2019.
- [19] J. Ost, F. Mannan, N. Thuerey, J. Knodt, and F. Heide, “Neural scene graphs for dynamic scenes,” in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2856–2865.
- [20] Z. Yang, Y. Chai, D. Anguelov, Y. Zhou, P. Sun, D. Erhan, S. Rafferty, and H. Kretzschmar, “Surfelgan: Synthesizing realistic sensor data for autonomous driving,” in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11118–11127.
- [21] I. Bozcan and E. Kayacan, “Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance,” in *IEEE Int. Conf. Robot. Autom.* IEEE, 2020, pp. 8504–8510.
- [22] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, “The unmanned aerial vehicle benchmark: Object detection and tracking,” in *Eur. Conf. Comput. Vis.*, 2018, pp. 370–386.
- [23] S. D. Morad, R. Mecca, R. P. Poudel, S. Liwicki, and R. Cipolla, “Embodied visual navigation with automatic curriculum learning in real environments,” *IEEE ROBOT. AUTOM. LET.*, vol. 6, no. 2, pp. 683–690, 2021.
- [24] J. Truong, S. Chernova, and D. Batra, “Bi-directional domain adaptation for sim2real transfer of embodied navigation agents,” *IEEE ROBOT. AUTOM. LET.*, vol. 6, no. 2, pp. 2634–2641, 2021.
- [25] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, “NeRF++: Analyzing and improving neural radiance fields,” *arXiv:2010.07492*, 2020.
- [26] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, “NeRF in the wild: Neural radiance fields for unconstrained photo collections,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2021, pp. 7210–7219.
- [27] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, “Block-NeRF: Scalable large scene neural view synthesis,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 8248–8258.
- [28] H. Turki, D. Ramanan, and M. Satyanarayanan, “Mega-NeRF: Scalable construction of large-scale nerfs for virtual fly-throughs,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 12922–12931.
- [29] M. Zhenxing and D. Xu, “Switch-NeRF: Learning scene decomposition with mixture of experts for large-scale neural radiance fields,” in *International Conference on Learning Representations*, 2022.
- [30] C. Sun, M. Sun, and H.-T. Chen, “Direct voxel grid optimization: Superfast convergence for radiance fields reconstruction,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 5459–5469.
- [31] Z. Wang, L. Li, Z. Shen, L. Shen, and L. Bo, “4k-NeRF: High fidelity neural radiance fields at ultra high resolutions,” *arXiv:2212.04701*, 2022.
- [32] J. Zhang, G. Yang, S. Tulsiani, and D. Ramanan, “Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild,” *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 29835–29847, 2021.
- [33] W. Jiang, V. Boominathan, and A. Veeraraghavan, “Nert: Implicit neural representations for unsupervised atmospheric turbulence mitigation,” in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 4235–4242.
- [34] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, “Building rome in a day,” *Commun. ACM*, vol. 54, no. 10, pp. 105–112, 2011.
- [35] C. Früh and A. Zakhov, “An automated method for large-scale, ground-based city model acquisition,” *Int. J. Comput. Vis.*, vol. 60, pp. 5–24, 2004.
- [36] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm, “Modeling and recognition of landmark image collections using iconic scene graphs,” in *Proc. Eur. Conf. Comput. Vis.* Springer, 2008, pp. 427–440.

- [37] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell *et al.*, “Detailed real-time urban 3d reconstruction from video,” *Int. J. Comput. Vis.*, vol. 78, pp. 143–167, 2008.
- [38] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 4104–4113.
- [39] N. Snavely, S. M. Seitz, and R. Szeliski, “Photo tourism: exploring photo collections in 3d,” in *SIGGRAPH*, 2006, pp. 835–846.
- [40] S. Zhu, R. Zhang, L. Zhou, T. Shen, T. Fang, P. Tan, and L. Quan, “Very large-scale global sfm by distributed motion averaging,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 4568–4577.
- [41] Z. Xie, X. Yang, Y. Yang, Q. Sun, Y. Jiang, H. Wang, Y. Cai, and M. Sun, “S3IM: Stochastic structural similarity and its unreasonable effectiveness for neural fields,” in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 18 024–18 034.
- [42] M. Ahmed, R. Seraj, and S. M. S. Islam, “The k-means algorithm: A comprehensive survey and performance evaluation,” *Electronics*, vol. 9, no. 8, p. 1295, 2020.
- [43] A. Fisher, R. Cannizzaro, M. Cochrane, C. Nagahawatte, and J. L. Palmer, “Colmap: A memory-efficient occupancy grid mapping framework,” *ROBOT AUTON SYST*, vol. 142, p. 103755, 2021.
- [44] Y. Xiangli, L. Xu, X. Pan, N. Zhao, A. Rao, C. Theobalt, B. Dai, and D. Lin, “BungeeNeRF: Progressive neural radiance field for extreme multi-scale scene rendering,” in *Eur. Conf. Comput. Vis.* Springer, 2022, pp. 106–122.
- [45] Y. Liu, F. Xue, and H. Huang, “Urbanscene3d: A large scale urban scene dataset and simulator,” *arXiv:2107.04286*, vol. 2, no. 3, 2021.
- [46] A. Hore and D. Ziou, “Image quality metrics: PSNR vs. SSIM,” in *Int. Conf. Pattern Recognit.* IEEE, 2010, pp. 2366–2369.
- [47] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.
- [48] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 586–595.
- [49] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv:1412.6980*, 2014.



Qi Liu is currently a Professor with the School of Future Technology at South China University of Technology. Dr. Liu received the Ph.D degree in Electrical Engineering from City University of Hong Kong, Hong Kong, China, in 2019. During 2018 - 2019, he was a Visiting Scholar at University of California Davis, CA, USA. From 2019 to 2022, he worked as a Research Fellow in the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include human-object interaction, AIGC, 3D scene reconstruction, and affective computing, etc. Dr. Liu has been an Associate Editor of the IEEE Systems Journal (2022-), and Digital Signal Processing (2022-). He was also Guest Editor for the IEEE Internet of Things Journal, IET Signal Processing, etc. He was the recipient of the Best Paper Award of IEEE ICSIDP in 2019.



Xiaohan Zhang received the M.Sc. degree with the School of Mathematics and Statistics, Shandong University, Weihai, China and received the B.S. degree from Xinjiang University, Urumqi, China. He is currently pursuing a doctor degree at the School of Future Technology, South China University of Technology. His major is Electronic Information. His research interest is 3D vision, including multi-view stereo, NeRF, GS and the application of 3D reconstruction technology in large-scale scenes.

Yukui Qiu is currently studying for a bachelor’s degree at South China University of Technology. His main research interest is computer vision.



Zhenyu Sun is currently working toward the bachelor’s degree with South China University of Technology. His research interests include computer vision.

