This ACCV 2024 paper, provided here by the Computer Vision Foundation, is the author-created version.
The content of this paper is identical to the content of the officially published ACCV 2024
LNCS version of the paper as available on SpringerLink: https://link.springer.com/conference/accv

Learning Neural Radiance Field from Quasi-Uniformly Sampled Spherical Image for Immersive Virtual Reality

Le Wang[1,2] and Shigang Li[✉1,3]

[1] Graduate School of Information Sciences, Hiroshima City University, Japan
[2] dg65001@e.hiroshima-cu.ac.jp
[3] shigangli@hiroshima-cu.ac.jp

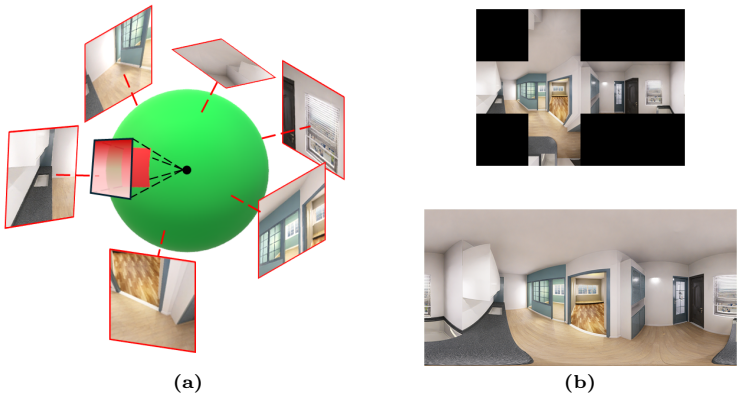
Abstract. The neural radiance field (NeRF) is a prominent method of novel view synthesis that is widely applied to many tasks. Recently, the NeRF method, which was originally developed for perspective images, has been extended to 360-degree omnidirectional images, which may make it easier to perform immersive virtual reality tasks. However, in existing NeRF methods, omnidirectional images are typically represented as rectangular images. Equirectangular image representation is popular but is also known for distortion, especially as it approaches the poles. In this paper, we propose a method for learning a neural radiance field from geodesic dome division-based discrete spherical images. First, an input equirectangular image is mapped to a unit sphere. Then, the mapped spherical image is sampled on a unit sphere on the basis of geodesic dome division. Next, the obtained discrete spherical image is used to train an NeRF network. Finally, comparative experiments are conducted for the equirectangular image representation and geodesic dome division discrete spherical image representation. The experimental results demonstrate that the proposed method outperforms existing equirectangular image representation methods.

Keywords: NeRF · Omnidirectional Image · Image Distortion · Geodesic Dome Division

1 Introduction

To realize immersive virtual reality, it is necessary to generate all of the surrounding scenes. This entails generating a 360-degree omnidirectional image. When an avatar moves in an environment, it is typically necessary to generate corresponding views continuously [1, 2]. If a new view can be synthesized from sparsely sampled images, the efficiency of the whole system is improved. In this work, we address the problem of synthesizing new omnidirectional images from a given omnidirectional image using color and distance information.

In the abovementioned applications, an omnidirectional image is a representation of environment maps. Perspective displays often need to be generated according to the gaze direction of an avatar. Ideally, the generated perspective



(a) (b)

Fig. 1: In VR, the system needs to synthesize the current screen according to the user's view, as shown in (a). (b) illustrates two representations of omnidirectional images: cubemap (first row) and equirectangular image (second row).

displays for any gaze direction of an avatar should have the same image quality, as shown in Fig. 1a. Because a perspective display is generated from an omnidirectional image (environment map), the sampling of the omnidirectional image used as an environment map on a sphere should be as uniform as possible.

Typically, an environment map is represented as a cubemap, which consists of six perspective images corresponding to the six faces of the circumscribed sphere or an equirectangular image [1–5], as shown in Fig. 1b.

The neural radiance field (NeRF) [6] is an advanced method for scene representation within a radiance field that has gained significant popularity. This approach infers scene information from images captured at various camera poses. Once trained, the NeRF model can synthesize novel images with photorealistic quality and continuous view changes in 3D scenes. Consequently, several studies have explored the integration of VR and NeRF, such as [7–9]. To improve NeRF performance on omnidirectional images, recent methods [10, 11] have extended NeRF from perspective cameras to omnidirectional cameras by using equirectangular images as inputs.

An equirectangular image is obtained by sampling a sphere along the latitude and longitude lines as if the sphere is Earth. As shown in Fig. 2a, the closer the poles are, the more densely the sphere is sampled. That is, there is a large sampling bias in this representation. Thus, if we were to use such a representation as the equirectangular image to learn a radiance field, the resulting radiance map would have redundant samples in the regions of the two poles compared to the region of the equator. Because some regions of the manifold of a radiance field are learned from many more samples than other regions, some regions may be approximated more accurately from a large number of samples than others from a small number of samples. Consequently, this variance influences the perspective images synthesized from the learned radiance field. That is, the quality of the

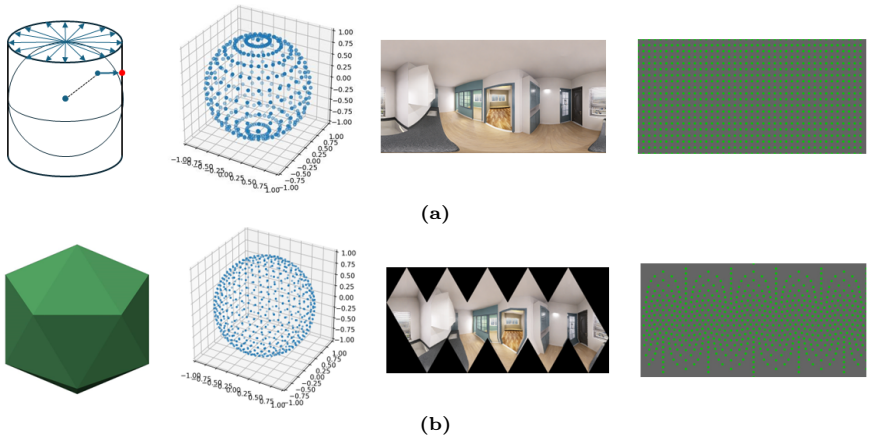


Fig. 2: The sampling methods and distributions for two representations: equirectangular image (a) and DSI (b). The first column depicts the sampling methods; the equirectangular image (a) projects the pixels of the sphere onto the surface of a cylinder, whereas the DSI (b) derives its sampling points from the vertices of a polyhedron that approximates the sphere. The second column shows the sampling distributions in a spherical camera model. It is apparent that DSI achieves a more uniform sampling distribution. The third column presents the RGB images for the two representations. The fourth column displays the distributions of the points sampled in the equirectangular image for both representations.

synthesized perspective images may change with the gaze direction of an avatar, which is not what we want

On the basis of the abovementioned observations, in this paper, we propose a method of learning a radiance field from a quasiuniformly sampled spherical image, that is, a geodesic dome division-based discrete spherical image (DSI), as shown in Fig. 2b. Because a regular polyhedron that has more than twenty regular faces cannot be obtained theoretically, the geodesic division of a sphere starts from an icosahedron, with each triangle divided into four smaller ones. In this work, we sample a sphere using this method [12]. The pixels of the obtained discrete spherical image correspond to the vertices of a geodesic dome. The resolution of the discrete spherical image is determined by the number of divisions. The discrete spherical image is represented as five connected parallelograms, which are obtained by flattening the divided icosahedron, as shown in Fig. 2b.

Note that the scene representation by the NeRF is learned from sample images. If we want to build a scene representation of the NeRF that approximates the truth, it is reasonable to learn the scene representation from as many uniform samples of the scene space as possible. Therefore, we argue that if we want to learn a scene representation from omnidirectional images, a uniformly sampled spherical image would be better than an omnidirectional image with heavy distortion, such as an equirectangular image.

In contrast with the existing methods of building radiance maps from omnidirectional images, such as OmniNeRF [10] and 360OmniNeRF [11], the contributions of this paper are as follows:

- We propose a method of building a radiance field of environments from quasiuniformly sampled spherical images. In contrast with the existing methods [10, 11], the problem of sampling bias of an environment map is eliminated. This also results in a more uniform radiance field.
- The proposed method is implemented for the recent neural radiance field (NeRF) approach. First, an input equirectangular image is mapped to a unit sphere. Then, the mapped spherical image is sampled on a unit sphere by using geodesic dome division. Next, the obtained discrete spherical image is used to train an NeRF network.
- We are the first to compare the performance of synthesizing perspective images with omnidirectional images in an NeRF, enabling a comparison of the influence of different image sampling distributions on NeRF model learning. Comparative experiments of radiance field buildings are conducted for equirectangular image representation and geodesic dome division discrete spherical image representation. The experimental results show that the proposed method outperforms the existing equirectangular image representation methods [10, 11].

The remainder of this paper is organized as follows: In Section 2, we introduce related work. In Section 3, we present the proposed method of learning the neural radiance field from geodesic dome division-based discrete spherical images. After the experimental results are presented in Section 4, the conclusions are presented in Section 5.

2 Related Work

In this section, we first briefly introduce the neural radiance field (NeRF) method and the progress that has been achieved recently. Next, we focus on related research using omnidirectional images.

2.1 Neural Radiance Field (NeRF) Method

The neural radiance field (NeRF) method was proposed in [6]. A continuous scene is represented as a 5D vector-valued function. Function F_Θ takes 3D location (x, y, z) and 2D viewing direction (θ, ϕ) as inputs and outputs an emitted color (r, g, b) and volume density σ, as follows:

$$(r, g, b, \sigma) = F_\Theta(x, y, z, \theta, \phi) \tag{1}$$

where Θ represents the parameters corresponding to the modeled scene. If function F_Θ is known, views can be synthesized by querying 5D coordinates along camera rays and using classic volume rendering techniques to project the

output colors and densities into an image. In [6], function F_Θ is learned as a neural network.

To date, many studies have been conducted on the NeRF. The seminal work by Mildenhall et al. [6] demonstrated the ability of an NeRF to render complex objects with high quality and high resolution via positional encoding [13] to embed input coordinates into a higher-frequency domain. After this study, numerous novel view synthesis methods using an NeRF have been proposed.

For example, D-NeRF [14] extends the NeRF framework to dynamic scenes, accounting for objects and lighting conditions that change over time and effectively capturing and rendering these variations. The NeRF in Wild [15] addresses challenges such as variable lighting, occlusion, and diverse scene content. FastNeRF [16] is a technique that renders high-fidelity photorealistic images at 200 Hz on high-end consumer GPUs. NeRF-Editing [17] introduces a method that allows users to perform controlled shape morphing on an implicit scene representation, synthesizing novel view images of the edited scene without retraining the network.

2.2 Learning the NeRF from omnidirectional images

As mentioned above, most of the studies on the NeRF learn the scene representation function F_Θ from perspective images. Some studies, such as [10] and [11], explore the application of an NeRF to omnidirectional images. OmniNeRF [10] uses one pair of RGB and depth omnidirectional images as scene data, projecting pixels onto the scene and moving a virtual camera to generate new views. 360OmniNeRF [11] incorporates attention and depth estimation modules to enhance NeRF learning. However, these omnidirectional NeRF methods employ an equirectangular image sampling distribution during training without addressing its limitations. Additionally, neither method has been tested on uniformly sampled perspective images.

In this paper, we propose a method of building NeRFs using quasiuniformly sampled spherical images. The NeRF network is trained on a discrete spherical image obtained via geodesic dome division. We argue that uniformly sampled spherical image training samples result in a better-learned surrounding scene representation, an NeRF for omnidirectional images. The proposed method is compared with previous similar methods [10] and [11] and outperforms the previous methods in a comparative experiment.

3 Method

We propose an omnidirectional NeRF method that utilizes discrete spherical images. The equirectangular image is first reprojected into the discrete spherical image format, as shown in Fig. 3a. Then, we use paired data $[RGB, depth]$ to map pixels to the spherical camera coordinate system. By moving the camera position, a training set is constructed, as shown in Fig. 3b. The NeRF is then trained on the pixels sampled from the scene. With the trained model, our framework is

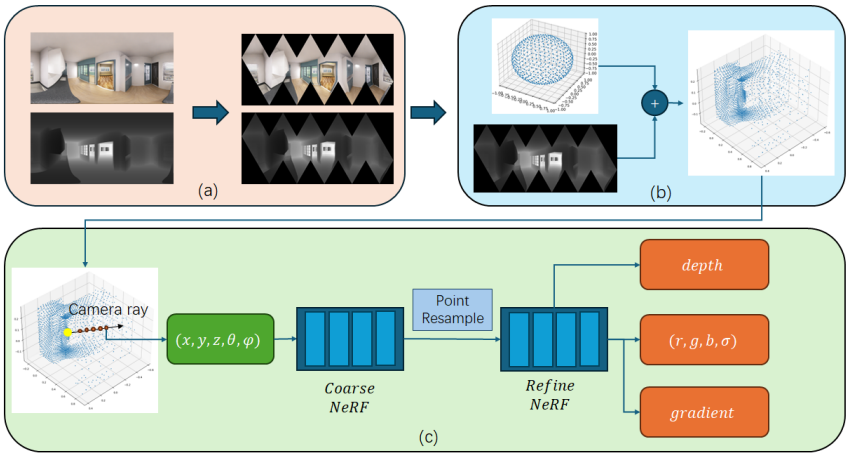


Fig. 3: First, a raw equirectangular image and depth map are projected as pixels onto the DSI image, as shown in (a). Next, images and depth information are used to project pixels onto the 3D camera coordinate system. The virtual camera is moved to the other position to get a novel view image for training, as shown in (b). Finally, 5D coordinates are fed into the NeRF, and then the RGB, volume density, depth, and gradient are estimated, as shown in (c).

capable of synthesizing three different representations of omnidirectional images: equirectangular images, discrete spherical images, and a cubemap. To evaluate the differences among these representations, we compared the performance of our method with OmniNeRF [10] and 360OmniNeRF [11]. Fig. 3 shows the flow of our proposed method.

3.1 Representations of omnidirectional image

In this work, we use two representations, equirectangular images and discrete spherical images, to train the NeRF model.

Equirectangular image Equirectangular image representation projects a spherical point $p_t(x, y, z)$ onto a point $p(v_0, u_0)$ on the surface of an external cylinder on the basis of its longitude and latitude, as shown in Fig. 2a. The point $p(v_0, u_0)$ is first converted to polar coordinates (θ, ϕ, r) via Eq. (2), where H and W denote the height and width of the equirectangular image, respectively.

$$\begin{aligned} \phi &= 2\pi(u_0 - W/2)/W \\ \theta &= \pi(v_0 - H/2)/H \end{aligned} \tag{2}$$

Discrete spherical image Discrete spherical images [12] are a technique for projecting omnidirectional images on the basis of geodesic dome division. This

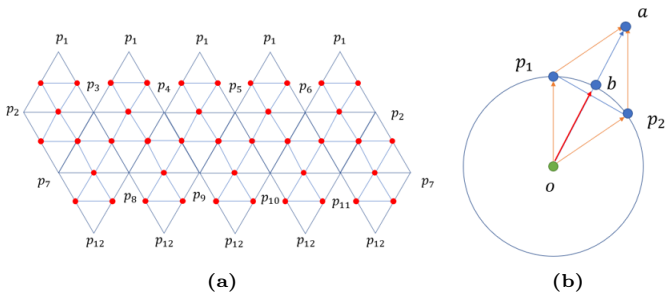


Fig. 4: (a) shows the vertex group (red points) obtained after one division. (b) shows the calculation method of the midpoint ob from adjacent vertices using vector synthesis.

method divides the triangular surfaces of an icosahedron by calculating the midpoints between adjacent vertices. The 12 vertices of the regular icosahedron are denoted as p_1 through p_{12}, as shown in Fig. 4a. The discrete spherical image involves calculating the midpoints between adjacent vertices and projecting these midpoints onto the unit sphere. The midpoint is calculated on the basis of the vectors and determined using Eq. (3). The first division result is indicated by the red points in Fig. 4a. With repeated divisions, the sampling points become dense, and the polyhedral model progressively approximates a sphere, as shown in Fig. 5.

$$\overrightarrow{ob} = (\overrightarrow{op_1} + \overrightarrow{op_2})/|\overrightarrow{op_1} + \overrightarrow{op_2}| \tag{3}$$

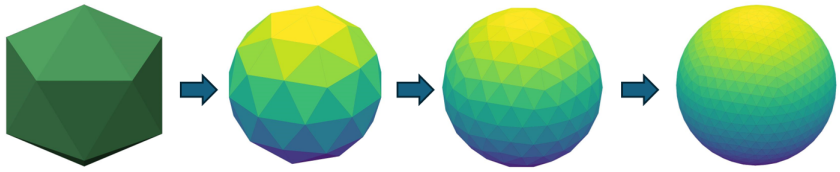


Fig. 5: The icosahedron is divided multiple times to obtain a model that approximates a sphere.

3.2 Generating dataset samples

The pixels on the equirectangular image are projected onto the spherical camera coordinate system via Eq. (2). Then, we use the division polyhedral model for sampling based on the vertices to generate discrete spherical images. We also sample the depth to obtain a depth map that matches the discrete spherical images, as shown in Fig. 3a.

Following the methods described in [10, 11], we create datasets for both equirectangular images and discrete spherical images. Our approach uses a paired omnidirectional image and depth map to generate novel view equirectangular images and discrete spherical images, as shown in Fig. 3b.

Specifically, the pixels are projected onto the camera coordinate system according to depth. By moving the camera and determining the sampling points of the emitted rays on the new sphere, we can generate new spherical images and depth maps. These generated images are used to train the network, with the original RGB images and depth maps serving as test data.

3.3 Network

The core component of our framework is the NeRF. The NeRF maps discrete samples of 5D coordinates, defined by spatial location (x, y, z) and viewing direction (θ, ϕ), to pixel color c_i and volume density σ_i via a multilayer perceptron (MLP), similar to the method described in [6]. To achieve better results, the input 5D points are encoded via a positional encoding technique [13]. The color composition function follows the rule on volume rendering [24]

$$C(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - \exp\left(-\sigma_i \delta_i\right)\right) \mathbf{c}_i, \tag{4}$$

where $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$, $\delta_i = t_{i+1} - t_i$ is the interval between two adjacent samples. To enable the network to focus on high-density areas, the NeRF network comprises two MLP networks, one "coarse" and one "refined", as shown in Fig. 3c. The color L_{color} loss between the estimated color and the ground truth is computed by using L_2 loss.

To enhance network performance in synthesizing novel view images, we incorporated two additional modules: depth and gradient. These modules help the NeRF perceive geometric and pixel-edge information.

Depth modules Inspired by [11], we utilized depth information to increase the credibility of the sampled point distribution, aiming to approximate the true geometric distribution more accurately. The depth calculation in our framework follows a process similar to color rendering [11]. The depth loss is the absolute difference between the predicted and true depths and is normalized by using the depth variance [18].

Gradient modules In this method, a single frame is used to train an entire scene, but the training dataset does not densely cover all regions. Consequently, when unseen areas are predicted, edges may appear blurred. To improve edge quality, we integrate a gradient prediction module inspired by [10]. This module uses a Laplacian filter to generate the ground truth. An additional output layer in our MLP model predicts the gradient of each sample point alongside the color output. The L_2 loss, $L_{gradient}$, is employed to optimize the gradient predictions.

3.4 Synthesizing different representation images

For a trained NeRF network, we can input 5D coordinates and calculate the color and opacity by sampling the corresponding ray in the scene model. As illustrated in Fig. 6a, the blue cube represents the scene information encoded by the NeRF. By sampling every pixel of the image, we can synthesize a complete image. Therefore, we can input different sampling distributions into the NeRF to obtain the desired representation, as follows:

$$I_d = F_\theta(\gamma) \tag{5}$$

where F_θ is the trained NeRF model and $\gamma \in \{d_{equ}, d_{dsi}, d_{pers}\}$ is the distribution of the desired representation image $I_d \in \{I_{equ}, I_{dsi}, I_{pers}\}$. We define each distribution in the form of $d = [s_1, s_2, ..., s_n]$, where s represents the sampled 5D coordinate system, I_{equ} is an equirectangular image, I_{dsi} is a discrete spherical image, and I_{pers} is a perspective image.

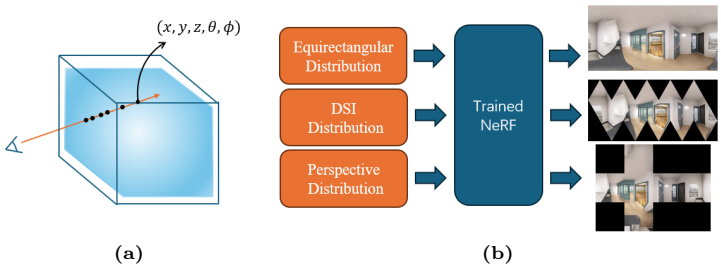


Fig. 6: (a) shows inputting 5D coordinates and calculating the color and opacity of the points by sampling the corresponding ray in the scene model. (b) shows that given different sampling distributions, the trained NeRF can synthesize corresponding representations.

To analyze the performance of the NeRF comprehensively, we synthesized three representations from each trained NeRF network, including an equirectangular image, a discrete spherical image, and a perspective image, as shown in Fig. 6b. The perspective images are merged into a cubemap. This method evaluates the models using sampling distributions not included in their training sets, allowing for a detailed comparison among them.

4 Experiment

4.1 Dataset

In our experiments, we used the Structured3D [19] dataset for training and testing. For testing, the following three representations were employed: an equirectangular image ((512, 1024) pixels), a discrete spherical image ((513, 1285) pixels),

and a cubemap image (each side is resolved at $(300, 300)$ pixels). The raw images and depth map were used to project pixels onto the 3D camera coordinate system. By moving the virtual camera, we generated 100 pairs of RGB and depth images with novel viewpoints to train the network. The raw image from the dataset served as the test data.

4.2 Implementation details

The network model was trained for 200,000 epochs by using a GTX 3090 Ti GPU . A batch size of 1,400 was employed during training. The Adam optimizer [20] was utilized with an initial learning rate of $5 \cdot 10^{-4}$, which was gradually reduced to $5 \cdot 10^{-5}$. According to our experimental framework, three networks were evaluated, as summarized in Tab. 1. "Omni" and "360" correspond to the models described in [10] and [11], respectively. The "Fusion" network integrates depth and gradient modules for enhanced performance. In the training setup, the parameters Coarse-Sample N_c and Refine-Sample N_f are set to 64 and 128, respectively. The quantitative results are evaluated by using PSNR, SSIM [21], and LPIPS [22, 23] to assess the quality of synthesized images.

Table 1: Network Configuration

Method	Gradient	Depth	Coarse-Sample	Refine-Sample
Omni	$\surd$	$\times$	64	128
360	$\times$	$\surd$	64	128
Fusion	$\surd$	$\surd$	64	128

4.3 Equirectangular image vs Discrete spherical images

To compare the differences between equirectangular images and discrete spherical images, we selected two scenes, "S1" and "S2", from the Structured3D [19] dataset for testing. We trained three networks, with each one using either equirectangular images (Equ) or discrete spherical images (DSI) as training data. The test set representation for each network matches the representation on which it was trained. The comparative test results are summarized in Tab. 2. Networks were trained by using discrete spherical images for our method.

In the "Method" column, the first word indicates the type of network, and the second word represents the data representation used in training. Better performance is indicated in bold. The metrics demonstrate that networks using discrete spherical images outperform those using equirectangular images. Using discrete spherical images enables the network to learn scene features more fully.

Table 2: Comparison of Equirectangular Images and Discrete Spherical Images

Method	S1			S2		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Omni Equ	31.7621	0.9052	0.2075	29.4672	0.8640	0.2185
Omni DSI	**35.1533**	**0.9451**	**0.1233**	**31.1774**	**0.9237**	**0.1421**
360 Equ	32.2467	0.9199	0.1910	29.6710	0.8956	0.2039
360 DSI	**35.4597**	**0.9476**	**0.1191**	**31.2880**	**0.9269**	**0.1347**
Fusion Equ	31.4689	0.8997	0.2137	29.5548	0.8861	0.2155
Fusion DSI	**35.1191**	**0.9443**	**0.1224**	**31.3847**	**0.9251**	**0.1405**

4.4 Comparison in different representation

This section is focused on evaluating the models by using sampling distributions that are not included in their training sets, allowing for a detailed comparison between them. We trained networks on two representations: equirectangular images and discrete spherical images. We then evaluated each network's performance by synthesizing equirectangular images, discrete spherical images, and cubemap images. The sampling distributions from discrete spherical images and perspective images are hidden when the networks are trained on equirectangular images. Conversely, the sampling distributions from equirectangular images and perspective images are hidden when the networks are trained on discrete spherical images. The results are detailed in Tabs. 3 to 5.

Table 3: Comparison of Synthesizing Equirectangular Images

Method	S1			S2		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Omni Equ	**31.7621**	**0.9052**	**0.2075**	**29.4672**	**0.8640**	**0.2185**
Omni DSI	30.8890	0.8818	0.2349	28.6312	0.8698	0.2346
360 Equ	**32.2467**	**0.9199**	**0.1910**	**29.6710**	**0.8956**	**0.2039**
360 DSI	31.1121	0.8837	0.2332	28.6106	0.8719	0.2313
Fusion Equ	**31.4689**	**0.8997**	**0.2137**	**29.5548**	**0.8861**	**0.2155**
Fusion DSI	30.7071	0.8811	0.2361	28.6975	0.8707	0.2323

According to the results presented in Tab. 3, networks trained with discrete spherical images do not perform as well as networks trained with equirectangular images when synthesizing nonuniform representations. Conversely, when synthesizing uniformly distributed discrete spherical images Tab. 4, networks trained with equirectangular images are less effective than networks trained with discrete spherical images. The results prove that the NeRF is sensitive to sampling distribution during training. The nonuniform distribution of sampling in equirectangular images can lead to the NeRF learning nonuniform scene features.

Table 4: Comparison of Synthesizing DSI Images

Method	S1			S2		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Omni Equ	31.6289	0.9303	0.1375	29.0027	0.8990	0.1612
Omni DSI	**35.1533**	**0.9451**	**0.1233**	**31.1774**	**0.9237**	**0.1421**
360 Equ	31.6718	0.9419	**0.1190**	29.2300	0.9125	0.1367
360 DSI	**35.4597**	**0.9476**	0.1191	**31.2880**	**0.9269**	**0.1347**
Fusion Equ	31.4483	0.9280	0.1397	29.1295	0.9011	0.1569
Fusion DSI	**35.1191**	**0.9443**	**0.1224**	**31.3847**	**0.9251**	**0.1405**

Table 5: Comparison of Synthesizing Cubemap Images

Method	S1			S2		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Omni Equ	34.7808	0.9666	0.07167	32.0983	0.9488	0.08464
Omni DSI	**37.9820**	**0.9735**	**0.06200**	**33.7191**	**0.9583**	**0.07504**
360 Equ	34.8285	0.9720	0.06328	32.3235	0.9553	0.07269
360 DSI	**38.3269**	**0.9749**	**0.06138**	**33.8370**	**0.9607**	**0.07163**
Fusion Equ	34.5876	0.9653	0.07257	32.2595	0.9497	0.08229
Fusion DSI	**37.9713**	**0.9731**	**0.06183**	**33.9052**	**0.9592**	**0.07316**

Moreover, when synthesizing cubemap images Tab. 5, networks trained with discrete spherical images achieve superior results. This suggests that it is difficult for the network to obtain uniform sampling with high accuracy from a network trained with a nonuniform sampling distribution.

In this experimental comparison, networks trained with equirectangular images have poorer performance when synthesizing discrete spherical images and cubemap images than networks trained with discrete spherical images. As mentioned above, to build a scene representation of an NeRF that approximates the truth, it is reasonable to learn the scene representation from as many uniform samples of the scene space as possible.

Overall, for an omnidirectional NeRF, the use of discrete spherical images offers advantages over equirectangular images. As depicted in Fig. 7, the cubemap synthesized by various networks demonstrates that the network using discrete spherical images effectively addresses issues encountered in training with equirectangular images.

5 Conclusion

The neural radiance field (NeRF) is learned from images in the training set. The nonuniform distribution in equirectangular images results in a network learned from nonuniformly distributed samples because of the sampling bias of an equirectangular image on a sphere. To address this issue, we propose a method

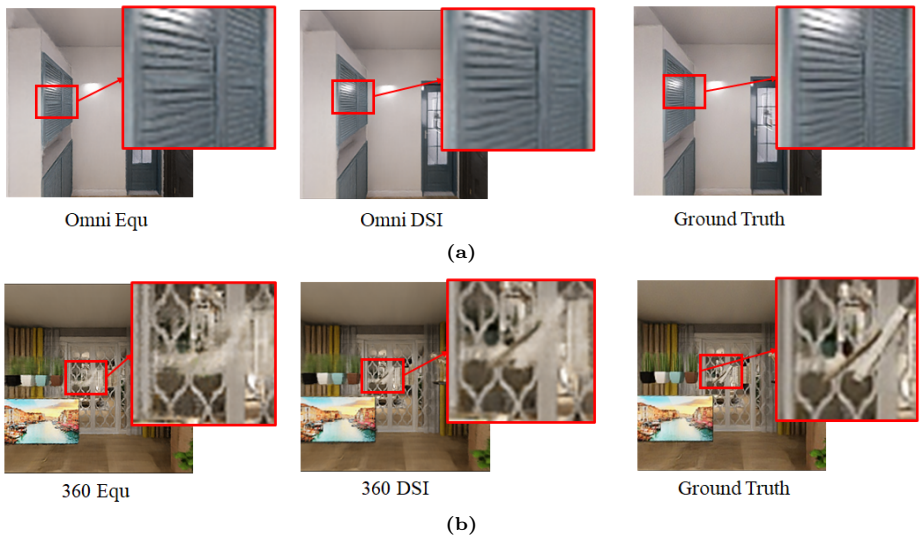


Fig. 7: A visualization of a novel view cubemap image. It is easy to see network trained with discrete spherical images fixes some pixel issues in the image. (a) shows the "Omni" network trained using equirectangular image (left) and discrete spherical images (center), synthesizing a cubemap of the "S1" scene. (b) shows the "360" network trained using equirectangular image (left) and discrete spherical images (center), synthesizing a cubemap of the "S2" scene.

of using quasiuniformly sampled discrete spherical images instead of equirectangular images as the training data for an omnidirectional NeRF. We conducted experiments on the Structured3D [19] dataset, comparing the performance of NeRF models with three types of distributions: equirectangular images, discrete spherical images, and perspective images. The experimental results demonstrate the effectiveness of the proposed method. The proposed method is expected to have applications in synthesizing gaze-point-free perspective images from an omnidirectional NeRF for the realization of immersive virtual reality.

Acknowledgments The work was supported by SATAKE Technology Promotion Foundation 2024 University Research Grant.

References

1. Google. (2024, July). Explore Street View and add your own 360 images to Google Maps. https://www.google.com/streetview/.
2. Li Y J, Shi J, Zhang F L, et al. Bullet comments for 360 video[C]//2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). IEEE, 2022: 1-10.
3. Chalmers A, Zaman F, Rhee T. Avatar360: Emulating 6-DoF Perception in 360° Panoramas through Avatar-Assisted Navigation[C]//2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR). IEEE, 2024: 630-638.

4. Ai H, Cao Z, Lu H, et al. Dream360: Diverse and Immersive Outdoor Virtual Scene Creation via Transformer-Based 360° Image Outpainting[J]. IEEE Transactions on Visualization and Computer Graphics, 2024.
5. Vermast A, Hürst W. Introducing 3d thumbnails to access 360-degree videos in virtual reality[J]. IEEE Transactions on Visualization and Computer Graphics, 2023, 29(5): 2547-2556.
6. Mildenhall B, Srinivasan P P, Tancik M, et al. Nerf: Representing scenes as neural radiance fields for view synthesis[J]. Communications of the ACM, 2021, 65(1): 99-106.
7. Deng N, He Z, Ye J, et al. Fov-nerf: Foveated neural radiance fields for virtual reality[J]. IEEE Transactions on Visualization and Computer Graphics, 2022, 28(11): 3854-3864.
8. Li C, Li S, Zhao Y, et al. Rt-nerf: Real-time on-device neural radiance fields towards immersive ar/vr rendering[C]//Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design. 2022: 1-9.
9. Xu L, Agrawal V, Laney W, et al. VR-NeRF: High-fidelity virtualized walkable spaces[C]//SIGGRAPH Asia 2023 Conference Papers. 2023: 1-12.
10. Hsu C Y, Sun C, Chen H T. Moving in a 360 world: Synthesizing panoramic parallaxes from a single panorama[J]. arXiv preprint arXiv:2106.10859, 2021.
11. Kulkarni S, Yin P, Scherer S. 360fusionnerf: Panoramic neural radiance fields with joint guidance[C], 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2023: 7202-7209.
12. Li S, Hai Y. A full-view spherical image format[C]//2010 20th International Conference on Pattern Recognition. IEEE, 2010: 2337-2340.
13. Tancik M, Srinivasan P, Mildenhall B, et al. Fourier features let networks learn high frequency functions in low dimensional domains[J]. Advances in neural information processing systems, 2020, 33: 7537-7547.
14. Pumarola A, Corona E, Pons-Moll G, et al. D-nerf: Neural radiance fields for dynamic scenes[C], Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 10318-10327.
15. Martin-Brualla R, Radwan N, Sajjadi M S M, et al. Nerf in the wild: Neural radiance fields for unconstrained photo collections[C], Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 7210-7219.
16. Garbin S J, Kowalski M, Johnson M, et al. Fastnerf: High-fidelity neural rendering at 200fps[C], Proceedings of the IEEE/CVF international conference on computer vision. 2021: 14346-14355.
17. Yuan Y J, Sun Y T, Lai Y K, et al. Nerf-editing: geometry editing of neural radiance fields[C], Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 18353-18364.
18. Sucar E, Liu S, Ortiz J, et al. imap: Implicit mapping and positioning in real-time[C], Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 6229-6238.
19. Zheng J, Zhang J, Li J, et al. Structured3d: A large photo-realistic dataset for structured 3d modeling[C], Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16. Springer International Publishing, 2020: 519-535.
20. Diederik P K. Adam: A method for stochastic optimization[J]. 2014.
21. Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to structural similarity[J]. IEEE transactions on image processing, 2004, 13(4): 600-612.

22. Zhang R, Isola P, Efros A A, et al. The unreasonable effectiveness of deep features as a perceptual metric[C], Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 586-595.
23. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
24. Max N. Optical models for direct volume rendering[J]. IEEE Transactions on Visualization and Computer Graphics, 1995, 1(2): 99-108.