Persistent Nature: A Generative Model of Unbounded 3D Worlds

Lucy Chai[1], Richard Tucker[2], Zhengqi Li[2], Phillip Isola[1], Noah Snavely[23]
[1]MIT [2]Google Research [3]Cornell Tech

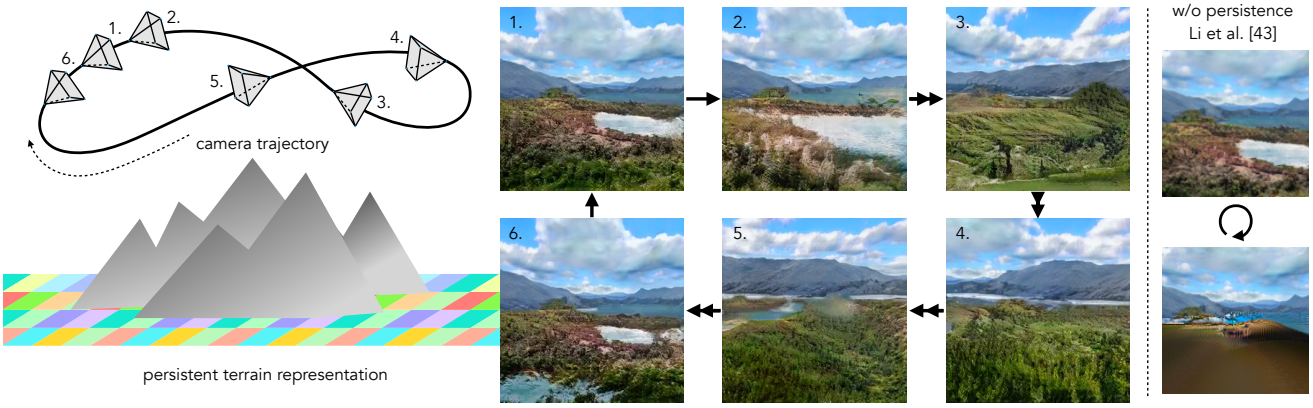


Figure 1. Our approach enables unconditional synthesis of unbounded 3D nature scenes with a persistent scene representation (**left**), using a scene layout grid representing a large-scale terrain model (depicted above as the checkered ground plane). This representation enables us to generate arbitrary camera trajectories, such as the six numbered views shown along a cyclic camera path (**center**). The *persistence* inherent to our representation stands in contrast to prior auto-regressive methods [43] that do not preserve consistency under circular camera trajectories (**right**); while the two images shown on the right are at the start and end of a cyclic path, the terrain depicted is completely different. Our method is trained solely from unposed, single-view landscape photos.

Abstract

Despite increasingly realistic image quality, recent 3D image generative models often operate on 3D volumes of fixed extent with limited camera motions. We investigate the task of unconditionally synthesizing unbounded *nature scenes, enabling arbitrarily large camera motion while maintaining a persistent 3D world model. Our scene representation consists of an extendable, planar scene layout grid, which can be rendered from arbitrary camera poses via a 3D decoder and volume rendering, and a panoramic skydome. Based on this representation, we learn a generative world model solely from single-view internet photos. Our method enables simulating long flights through 3D landscapes, while maintaining global scene consistency—for instance, returning to the starting point yields the same view of the scene. Our approach enables scene extrapolation beyond the fixed bounds of current 3D generative models, while also supporting a persistent, camera-independent world representation that stands in contrast to auto-regressive 3D prediction models. Our project page:* https://chail.github.io/persistent-nature/.

1. Introduction

Generative image and video models have achieved remarkable levels of realism, but are still far from presenting a convincing, explorable world. Moving a virtual camera through these models—either in their latent space [4, 24, 30, 74] or via explicit conditioning [36]—is not like walking about in the real world. Movement is either very limited (for example, in object-centric models [6]), or else camera motion is unlimited but quickly reveals the lack of a persistent world model. Auto-regressive 3D synthesis methods exemplify this lack of persistence [43, 46]; parts of the scene may change unexpectedly as the camera moves, and you may find that the scene is entirely different when returning to previous positions. The lack of spatial and temporal consistency can give the output of these models a strange, dream-like quality. In contrast, machines that can generate unbounded, persistent 3D worlds could be used to develop agents that plan within a world model [22], or to build virtual reality experiences that feel closer to the natural world, rather than appearing as ephemeral hallucinations [43].

We therefore aim to develop a unconditional generative model capable of generating unbounded 3D scenes with a

persistent underlying world representation. We want synthesized content to move in a way that is consistent with camera motion, yet we should also be able to move arbitrarily far and still generate the same scene upon returning to a previous camera location, regardless of the camera trajectory.

To achieve this goal, we model a 3D world as a *terrain* plus a *skydome*. The terrain is represented by a *scene layout grid*—an extendable 2D array of feature vectors that acts as a map of the landscape. We 'lift' these features into 3D and decode them with an MLP into a radiance field for volume rendering. The rendered terrain images are super-resolved and composited with renderings from the skydome model to synthesize final images. We train using a layout grid of limited size, but can extend the scene layout grid by any desired amount during inference, enabling unbounded camera trajectories. Since our underlying representation is persistent over space and time, we can fly around 3D landscapes in a consistent manner. Our method does not require multiview data; each part of our system is trained from an unposed collection of single-view images using GAN objectives.

Our work builds upon two prior threads of research that tackle generating immersive worlds: 1) generative models of 3D data, and 2) generative models of infinite videos. Along the first direction are generators of meshes, volumes, radiance fields, etc (e.g., [6, 56, 61]). These models represent a consistent 3D world by construction, and excel at rendering isolated objects and bounded indoor scenes. Our work, in contrast, tackles the challenging problem of generating large-scale *unbounded* nature scenes. Along the second direction are methods like InfiniteNature [43, 46], which can indeed simulate visual worlds of infinite extent. These methods enable unbounded scene synthesis by predicting new viewpoints auto-regressively from a starting view. However, they do not ensure a persistent world representation; content may change when revisited.

Our method aims to combine the best of both worlds, generating boundless scenes (unlike prior 3D generators) while still representing a persistent 3D world (unlike prior video generative models). In summary:

- We present an unconditional 3D generative model for unbounded nature scenes with a persistent world representation, consisting of a terrain map and skydome.
- We augment our generative pipeline to support camera extrapolation beyond the training camera distribution by extending the terrain features.
- Our model is learned entirely from single-view landscape photos with unknown camera poses.

2. Related Work

Image and view extrapolation. Pioneering work by Kaneva *et al*. [33] proposed the task of infinite image extrapolation by using a large image database to perform classical 2D image retrieval, stitching, and rendering. More recently, various learning-based 2D image inpainting [25, 42, 47, 69, 81, 96, 97, 99] and outpainting [3, 10, 44, 82, 91, 93] methods have been developed. These methods fill in missing image regions or expand the field of view by synthesizing realistic image content that is coherent with the partial input image. Beyond 2D, prior work has explored single-view 3D *view extrapolation*, often by applying 2D image synthesis techniques within a 3D representation [29, 31, 41, 67, 68, 76, 92]. However, these methods can only extrapolate content within a very limited range of viewpoints.

Video generation. Video generation aims to synthesize realistic videos from different types of input. Unconditional video generation produces long videos often from noise input [4, 18, 20, 48, 55, 79, 85], while conditional video generation generates sequences by conditioning on one or a few images [14, 16, 28, 37, 39, 86, 87, 87, 88, 90, 94, 98], or a text prompt [27, 77]. However, applying these ideas in 3D requires supervision from multi-view training data, and cannot achieve persistent 3D scene content at runtime, since there is no explicit 3D representation. Some recent work preserves global scene consistency via extra 3D geometry inputs such as point clouds [50] or voxel grids [23]. In contrast, our method synthesizes both the geometry and appearance of an entire world from scratch using a global feature representation to achieve consistent generated content.

Generative view synthesis. Novel view synthesis aims to produce new views of a scene from single [8, 32, 38, 59, 68, 75, 76, 83, 84, 92, 95] or multiple image observations [2, 11, 17, 40, 49, 52–54, 66, 70, 73, 89, 100] by constructing a local or global 3D scene representation. However, most prior methods can only interpolate or extrapolate a limited distance from the input views, and do not possess a generative ability.

On the other hand, a number of generative view synthesis methods have been recently proposed utilizing neural volumetric representations [6, 15, 21, 56–58, 64, 71, 80]. These methods can learn to generate 3D representations from 2D supervision, and have demonstrated impressive results on generating novel objects [61], faces [6, 13, 21, 60], or indoor environments [15, 65]. However, none of these methods can generate unbounded outdoor scenes due to lack of multi-view data for supervision, and due to the larger and more complex scene geometry and appearance that is difficult to model with prior representations. In contrast, our approach can generate globally consistent, large-scale nature scenes by training solely from unstructured 2D photo collections.

Our work is particularly inspired by recent perpetual view generation methods, including InfiniteNature [46] and InfiniteNature-Zero [43], which can generate unbounded flythrough videos of natural scenes, and are trained on nature videos or photo collections. However, these methods generate video sequences in an auto-regressive manner, and therefore cannot achieve globally consistent 3D scene content.

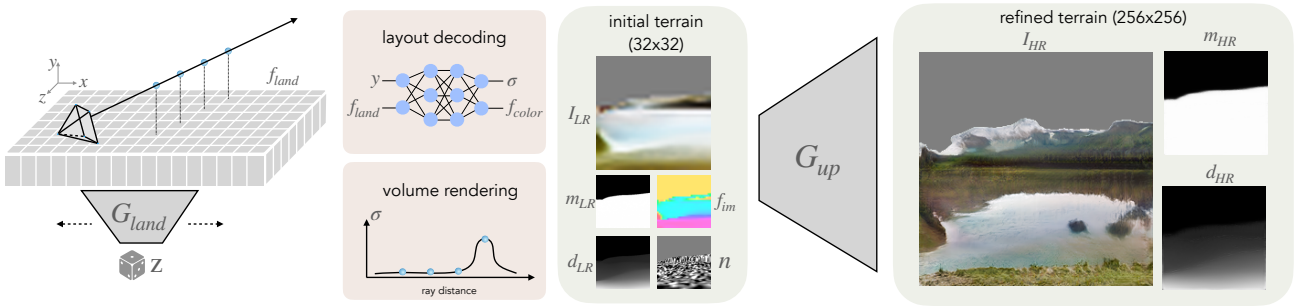


Figure 2. *Overview of scene layout decoding.* The layout generator G_{land} samples a random latent code to produce a 2D scene layout grid f_{land} representing the shape and appearance of a terrain map, and which can be spatially extended using a grid of latent codes (see § 3.2). To render an image from a given camera, sampled points along camera rays passing over the feature plane are decoded via an MLP into a color feature f_{color} and density σ, which are then volume rendered. This produces a low-resolution image, mask, depth, image features, and a projected noise pattern, which are provided to a refinement network G_{up} to produce final image, mask, and depth outputs.

Our approach instead adopts a global scene representation that can be trained to generate consistent-by-construction and realistic novel views spanning large-scale scenes. Concurrent works for scene synthesis InfiniCity [45] and SceneDreamer [9] leverage birds-eye-view representations, while SceneScape [19] builds a mesh representation from text.

3. Method

Our scene representation for unbounded landscapes consists of two components, a *scene layout grid* and a *skydome*. The scene layout grid models the landscape terrain, and is a 2D grid of features defined on a "ground plane." These 2D features are intended to describe both the height and appearance content of the terrain, representing the full 3D scene — in fact, we decode these features to a 3D radiance field, which can then be rendered to an image (§3.1). To enable camera motion beyond the training volume, we spatially extend the 2D feature grid to arbitrary sizes (§3.2). Because it is computationally expensive to generate and volume render highly detailed 3D content at the scale we aim for, we use an image-space refinement network that adds additional texture detail to rendered images (§3.3).

The second scene component is a *skydome* (§3.4), which is a spherical (panoramic) image intended to model very remote content, such as the sun and sky, as well as distant mountains. The skydome is generated to harmonize with the terrain content described by the scene layout grid.

All the stages of our approach are trained with GAN losses (§3.5). In what follows, we use the 3D coordinate convention that the ground plane is the xz-plane, and the y-axis represents height above or below this plane. Generally, the camera used to view the scene will be positioned some height above the ground.

3.1. Scene layout generation and rendering

To represent a distribution over landscapes, we take a generative approach following the layout representation of GSN [15]. First, a 2D scene layout grid is synthesized from a sampled random noise code $\mathbf{z}$ passed to a StyleGAN2 [35] generator G_{land}. This creates a 2D feature grid f_{land}, which we bilinearly interpolate to obtain a 2D function over spatial coordinates x and z:

$$f_{\text{land}}(x, z) = \text{Interpolate}(G_{\text{land}}(\mathbf{z}), (x, z)) \tag{1}$$

To define a full 3D scene, we need a way to compute the content at any 3D location (x, y, z). We define a multi-layer perceptron M that takes a scene grid feature, as well as the height y of the point at which we want to evaluate the scene content. The outputs of M are the 2D-to-3D lifted feature f_{color} and the density σ at point (x, y, z):

$$f_{\text{color}}, \sigma = M(f_{\text{land}}(x, z), y). \tag{2}$$

In this way, the 2D scene layout grid determines a radiance field over all 3D points within the bounds of the grid [15, 72, 95]. That is, feature vectors in the grid encode not just appearance information, but also the height (or possibly multiple heights) of the terrain at their ground location.

To render an image from a desired camera pose, we cast rays $\mathbf{r}$ from the camera origin through 3D space, sample points (x, y, z) along them, and compute f_{color} and σ at each point. We then use volume rendering to composite f_{color} along each ray into projected 2D image features f_{im}, a disparity image d_{LR}, and a sky segmentation mask m_{LR}. We form an initial RGB image of the terrain, I_{LR}, via a learned linear projection P of these image features. This process is depicted in the left half of Fig. 2, and is defined as:

$$\begin{aligned} f_{\text{im}}(\mathbf{r}) &= \sum_{i=1}^{N} w_i f_{\text{color},i}, \quad & d_{\text{LR}}(\mathbf{r}) &= \sum_{i=1}^{N} w_i d_i, \\ m_{\text{LR}}(\mathbf{r}) &= \sum_{i=1}^{N} w_i, & I_{\text{LR}} &= P f_{\text{im}}, \end{aligned} \tag{3}$$

where $i \in \{1..N\}$ refers to the index of each sampled point along ray $\mathbf{r}$ in order of increasing distance from the camera,

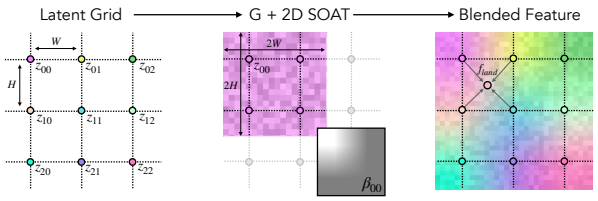


Figure 3. *Layout extension procedure.* To extend the layout at inference time, we sample noise codes $\mathbf{z}$ in a grid arrangement. To smoothly transition between adjacent feature grids, we use the SOAT (StyleGAN of All Trades) procedure [12] in 2D. Operating on a 2×2 sub-grid, we apply each generator layer four times in fully convolutional manner over the entire sub-grid, each time conditioned on a different corner latent code $\mathbf{z}$, before multiplying by bilinear blending weights. This process is repeated for each layer of the generator and each sub-grid. Each 2×2 sub-grid produces a $2H \times 2W$ feature grid, and sub-grids are blended together in an overlapping fashion to obtain an extended feature grid f_{land} of arbitrary spatial size.

d_i is the inverse-depth (disparity) of point i, and weights w_i are determined from the volume rendering equations used in NeRF [53] (see supplemental).

We intend the mask m_{LR} to distinguish sky regions (which will be empty and filled later using the skydome) from non-sky regions, and achieve this by training using segmented real images in which color and disparity for sky pixels are replaced with zero. Since to achieve zero disparity all weights along a ray must be zero (which also results in a zero-valued color feature), this approach encourages the generator to omit sky content. However, while we find that the model indeed learns to generate transparent sky regions, land geometry can also become partially transparent. To counter this, we penalize visible decreases in opacity along viewing rays using finite differences of opacity α:

$$\mathcal{L}_{\text{transparent}}(\mathbf{r}) = \sum_{i=2}^{N} w_i \frac{\max(\alpha_{i-1} - \alpha_i, 0)}{\delta_i}. \tag{4}$$

3.2. Layout Extension

While G_{land} creates a fixed-size feature grid, our objective is to generate geometry of arbitrary size, enabling long-distance camera motion at inference time. Hence, we devise a way to *extend* the feature grid in the x and z dimensions. We illustrate this process in Fig. 3, where we first sample noise codes $\mathbf{z}$ in a grid arrangement, where each $\mathbf{z}$ generates a 2D layout feature grid of size $H \times W$. To obtain a smooth transition between these independently sampled layout features, we generalize the image interpolation approach from SOAT (StyleGAN of all Trades) [12] to two dimensions. We operate on 2×2 sub-grids and blend intermediate features from each layer of the generator as follows:

$$\begin{aligned} f_{k,l+1} &= G_l(f_l, \mathbf{z}_k); \quad k = \{00,\ 01,\ 10,\ 11\} \\ f_{l+1} &= \sum_{k=\{00,01,10,11\}} \beta_k(x,z) f_{k,l+1}. \end{aligned} \tag{5}$$

For each of the four corner anchors k, we construct the modulated feature $f_{k,l+1}$ by applying G_l (the l-th layer of G_{land}) in a fully convolutional manner over the entire sub-grid. We then interpolate between the four feature grids using bilinear interpolation weights $\beta_k(x,z)$. By stitching these 2×2 sub-grids in an overlapping manner, we can obtain a scene layout feature grid of arbitrary size to use as f_{land}. Additional details are provided in the supplemental.

3.3. Image refinement

Due to the computational cost of volume rendering, training the layout generator at higher resolutions becomes impractical. We therefore use a refinement network G_{up} to upsample the initial generated image I_{LR} to a higher-resolution result I_{HR}, while adding textural details (Fig. 2-right). We use a StyleGAN2 backbone for G_{up}, replacing the earlier feature layers with feature output f_{im} and the RGB residual layers with a concatenation of I_{LR}, d_{LR}, and m_{LR}. To encourage the refined terrain image I_{HR} to be consistent with the sky mask, the network also predicts a refined disparity map and sky mask for compositing with the skydome (see §3.4):

$$I_{\text{HR}}, d_{\text{HR}}, m_{\text{HR}} = G_{\text{up}}(f_{\text{im}}, I_{\text{LR}}, d_{\text{LR}}, m_{\text{LR}}). \tag{6}$$

We compute a reconstruction loss between the initial and refined disparity and mask outputs, and penalize G_{up} for producing gray sky pixels in I_{HR} outside the predicted mask m_{HR}. Please see the supplemental for more details.

For fine texture details, StyleGAN2 also uses layer-wise spatial noise in intermediate generator layers (in addition to the global latent $\mathbf{z}$). Using a fixed 2D noise pattern results in texture 'sticking' as we move the camera [34], but resampling it every frame reduces spatial coherence and removing it entirely results in convolutional gridding artifacts. To avoid these issues and improve spatial consistency, we replace the 2D image-space noise with projected 3D world-space noise, where the noise input to G_{up} is the projection of samples from a grid of noise, n. This noise pattern is drawn from a standard Gaussian distribution defined on the ground plane at the same resolution of the layout features, which is then lifted into 3D and volume rendered along each ray $\mathbf{r}$:

$$n(\mathbf{r}) = \sum_{i=1}^{N} w_i n(x,z). \tag{7}$$

3.4. Skydome

We model remote content (sky and distant mountains) separately with a skydome generator G_{sky} (Fig.4). This generator follows the StyleGAN3 architecture [34], with a mapping

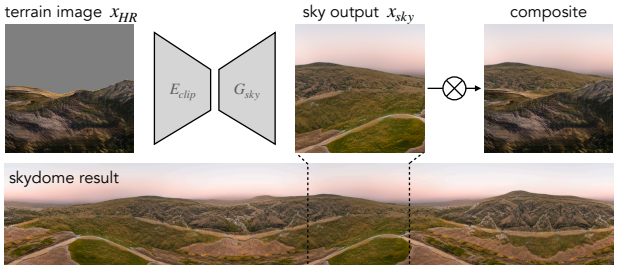


Figure 4. *Skydome generator.* Conditioned on the terrain image, the skydome generator G_{sky} synthesizes distant content (e.g., sky pixels and remote mountains) that is consistent with the generated terrain using encoder E_{clip}. G_{sky} is conditioned on cylindrical coordinates which can be unwrapped to produce a panoramic skydome image.

network and synthesis network conditioned on cylindrical coordinates [5]. We adapt it by conditioning on the terrain output: we encode terrain images I_{HR} using the pretrained CLIP image encoder E_{clip} [62], and concatenate this to the style-code output of the mapping network as input into G_{sky}:

$$I_{\text{sky}} = G_{\text{sky}}(\text{concat}(E_{\text{clip}}(I_{\text{HR}}), \text{mapping}(\mathbf{z}))). \quad (8)$$

Conditioning on the foreground terrain image encourages the skydome generator to generate a sky that is consistent with the terrain content. This model trains on single-view landscape images but can produce a full panorama at inference-time by passing in coordinates that correspond to a 360° cylinder. The skydome is rendered to an individual camera viewpoint using camera ray directions, giving the skydome image I_{dome} which is then composited with the terrain image using the sky mask:

$$I_{\text{full}} = I_{\text{HR}} \odot m_{\text{HR}} + I_{\text{dome}} \odot (1 - m_{\text{HR}}). \quad (9)$$

3.5. Training

We train the layout generator (rendering at 32x32), refinement network (upsampling to 256x256), and skydome generator separately. To train the refinement network, we operate on outputs of the layout generator, freezing the weights of that model. For the skydome generator, we train using real landscape images, and apply it only to the outputs of the refinement network at inference time. We follow the StyleGAN2 objective [35], with additional losses for each training stage, architecture, and hyperparameters provided in the supplemental.

Dataset and camera poses. We train on LHQ [78], a dataset of of 90K unposed, single-view images of natural landscapes. A number of LHQ images contain geometry that is not amenable to "flying", such as a landscape pictured through a window, or a closeup of trees. Therefore, we perform a filtering process on LHQ prior to training (see supplemental). We also obtain auxiliary outputs – disparity and sky segmentation – using the pretrained DPT [63] model. Disparity and sky segmentation are used to construct the real image distribution in the GAN training phases.

After filtering, we use 56,982 images for training, and augment with horizontal flipping. During training we also need to sample camera poses. Prior 3D generators [6, 7, 15, 21, 60, 71] either use ground-truth poses from a simulator, or assume an object-centric camera distribution in which the camera looks at a fixed origin from some radius. Because our dataset lacks ground truth poses, we first sample a bank of training poses uniformly across the layout feature grid with random small height offsets, and rotate such that the near half of the camera view frustum falls entirely within the layout grid. Since the aerial layout should not be specific to any given camera pose, we generate f_{land} without any camera pose information, and then adopt the sampling scheme from GSN [15] which samples a camera pose from the initial training pose bank proportional to the inverse terrain density at each camera position, to avoid placing the camera within occluding geometry.

4. Experiments

Given its persistent scene representation and the extensibility of the its layout grid, our model enables arbitrary motion through a synthesized landscape, including long camera trajectories. We show sample outputs from our model under a variety of camera movements (§ 4.1); present qualitative and quantitative comparisons with alternate scene representations, including auto-regressive prediction models and unconditional generators defined for bounded or object-centric scenes (§ 4.2); and investigate variations of our model to evaluate design decisions (§ 4.3).

4.1. Persistent, unbounded scene synthesis

Figure 5 shows example landscapes generated by our model with various camera motions. As the camera moves (by rotating and/or translating) the generated imagery changes in a way that is consistent with the underlying geometry, *e.g.* hills move across the image or become closer. Extending the generated aerial feature grid allows us to place the camera *outside* the distribution of training camera poses, while maintaining both geometric and stylistic consistency. As illustrated in Figure 1 and our project page, the persistent and extendable layout features enables synthetic 'flights' over large distances that can also return to a consistent starting point.

4.2. Comparing scene representations

We compare our model with three state-of-the-art methods. InfiniteNature-Zero is an auto-regressive method that, given an initial frame, generates successive frames sequentially by warping each image to the next based on depth [43]. It allows for unbounded camera trajectories, but has no persistent world model. GSN [15] and EG3D [6] are uncon-

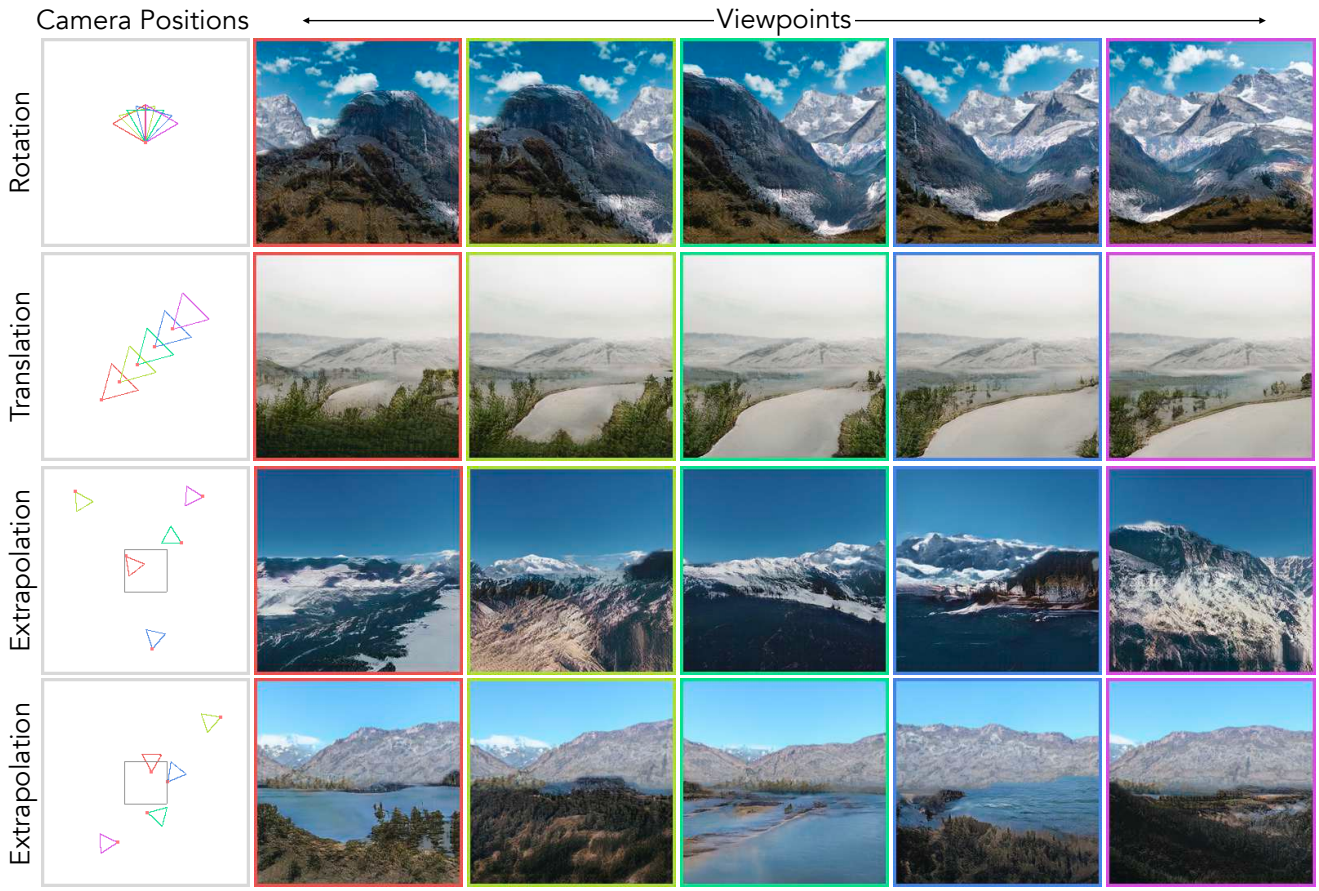


Figure 5. *Visualization of nearby and extrapolated camera motion.* Each row shows a set of sampled viewpoints, shown in an overhead view in the first column, and the corresponding rendered images in the other columns. Our model enables 3D-consistent view synthesis, visible under rotating or translating camera trajectories. We can also extrapolate the layout features at inference time, enabling camera motions outside of the training camera distribution (shown as a black square in the last two rows) with a consistent scene style.

Model	Persistent	Unbounded	FID	Consistency	
			C_{forward}	1-step	cycle
Inf Nat Zero [43]	✗	✓	28.15	**1.84**	3.94
Ours (128px)	✓	✓	**26.09**	2.12	**0.00**

Table 1. *Comparison with InfiniteNature-Zero.* Using camera motions from InfiniteNature-Zero, we evaluate image quality as FID on 5K images after moving 100 steps forward (C_{forward}), one-step consistency as the L1 error when backwards warping one camera step, and cycle consistency as the L1 error between the original frame and the result after a pair of forward/backward steps. InfiniteNature-Zero is more consistent for a single step, but it has non-zero cyclic consistency error, and image quality degrades after repeated model applications. L1 values are multiplied by 100 throughout.

ditional generative models: GSN uses a layout feature grid which is also the basis of our model, but focuses on bounded indoor scenes with ground-truth camera pose trajectories, while EG3D uses a tri-plane representation and primarily focuses on objects and portraits. These methods have persistent world models (feature grid and tri-plane representation) but do not allow for unbounded trajectories.

Model	Persistent	Unbounded	FID			Consistency
			C_{train}	C_{forward}	C_{random}	
GSN [15]	✓	✗	29.95	50.22	45.48	12.80
EG3D [6]	✓	✗	**9.85**	30.17	32.08	**3.01**
Ours	✓	✓	21.42	**26.67**	**23.39**	3.56

Table 2. *Quantitative comparison to unconditional GANs.* We evaluate image quality as FID on 5K images on (a) training camera poses C_{train}, (b) forward motion C_{forward} (See Table 1), (c) random camera poses C_{random}. One-step consistency error is measured as the L1 error when backwards warping the result after one camera step to the initial frame, multiplied by 100. Once outside the training pose distribution our model generates better images than other methods, with consistency close to that of EG3D.

Quantitative comparisons. We evaluate image quality using FID [26], and multi-view consistency using photometric error. To compare with InfiniteNature-Zero (Table 1), we initialize with an image and depth map from our model, move the camera forwards using a forward motion trajectory from InfiniteNature-Zero, and evaluate image quality at a distance of 100 forward steps. Our model attains better FID, showing that it does not suffer from image degradation due to succes-

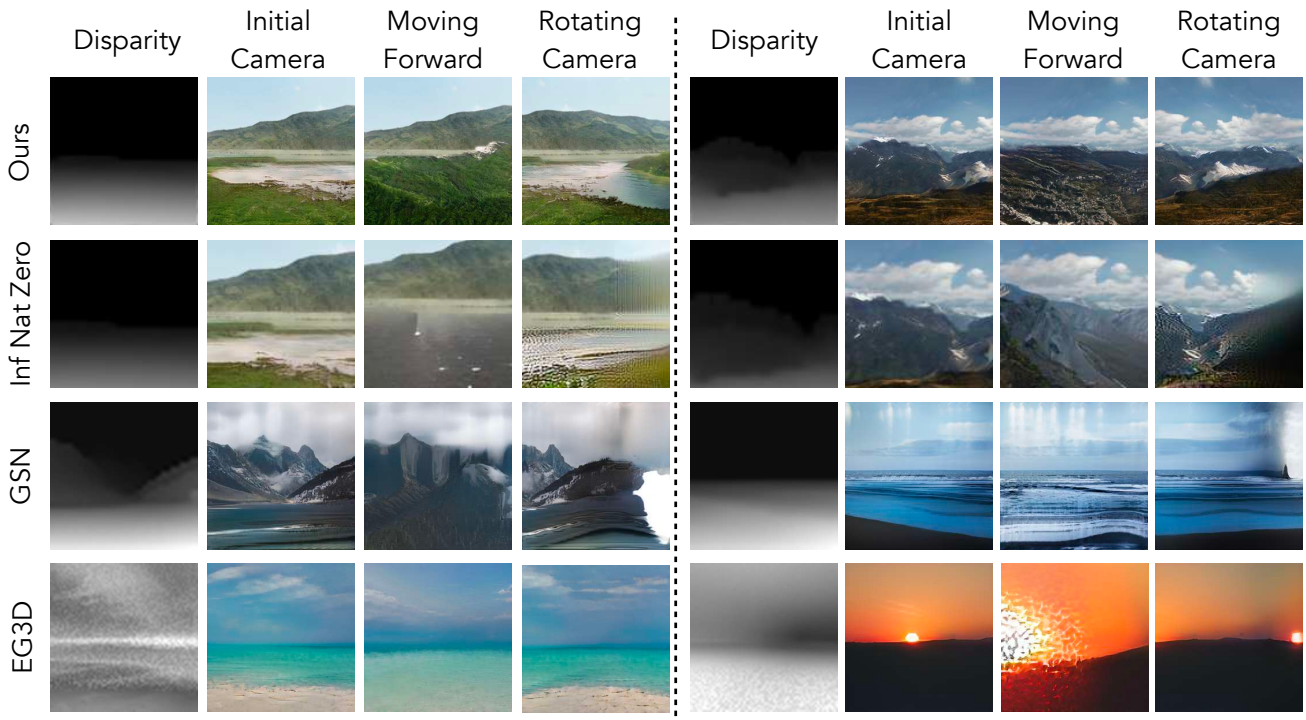


Figure 6. *Comparison to auto-regressive and bounded-volume 3D generative models.* Each row shows results for a given method on two generated scenes under different camera motion, along with a disparity map. Compared to InfiniteNature-Zero, our model enables long-range view synthesis by rendering a global scene description from different viewpoints, rather than auto-regressively predicting successive frames. 3D generative models like EG3D and GSN do not support view extrapolation on unbounded scenes. See our webpage for animated results.

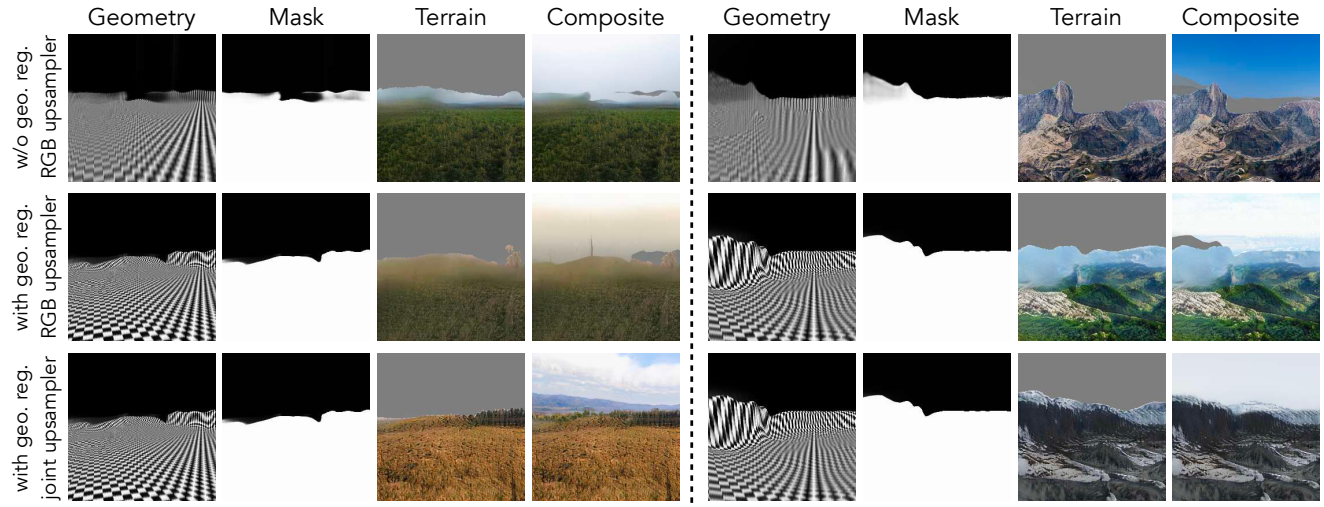


Figure 7. *Qualitative comparison of model variations.* Each row shows a model variant, visualizing generated geometry (as a rendered scene filled with a checkerboard pattern), sky mask, rendered terrain, and final image composite. (Top) Without geometry regularization, the model produces semi-transparent terrain. (Middle) Adding geometry regularization (Eqn. 4) makes the terrain more solid, but there are inconsistencies between the terrain and mask prediction. (Bottom) Our full model uses geometry regularization and also adds a upsampler that operates on inverse-depth and sky mask inputs in addition to RGB (Eqn. 6) to discourage boundary effects between the terrain and sky.

sive applications of an auto-regressive model. To compute one-step consistency error, we generate a new frame at a position equivalent to one forward step of InfiniteNature-Zero, warp it back to the original camera position using depth, and compute L1 error with the original frame in the overlapping region. Because InfiniteNature-Zero uses explicit warping as part of its model, it can achieve better one-step consistency, whereas our 2D upsampling operation is more susceptible to geometric inconsistency. We measure cyclic consistency error as the L1 error between the initial frame to the result

after a step forward and back. Because InfiniteNature-Zero lacks a persistent global representation, it has non-zero cyclic consistency error, whereas our model is fully consistent with zero cyclic consistency error.

To compare with the unconditional generative models GSN and EG3D, we compute FID on sets of output images corresponding to different distributions of camera positions: camera poses used in training which are intended to overlap with the layout, camera poses 100 steps forward from these mimicking InfiniteNature-Zero trajectories, and a uniform distribution of randomly oriented cameras over the layout grid. As seen in Table 2, GSN is the least successful method when applied to this domain. EG3D generates high-quality images at training camera poses, but tends to represent the scene as floating nearby clouds with planar mountains at the edges of the volume (incorrect geometry). Our method generalizes better to new camera positions. GSN has the highest one-step consistency error, while the consistency error of our model is close to that of EG3D (which relies less on 2D upsampling). In the supplemental, we experiment with an alternative architecture that builds on extendable triplane units with lower consistency error and faster rendering speed.

Qualitative comparisons. In Fig. 6 we show example outputs of each model over forward-moving and rotating trajectories. Due to its auto-regressive nature, the quality of InfiniteNature-Zero's output degrades somewhat as the camera trajectory becomes longer. A more serious limitation is that, trained only on forward movement, it is unable to synthesize plausible views under camera rotation. GSN and EG3D also struggle with long camera trajectories, producing unrealistic outputs as the cameras approach the spatial limits of the training camera distribution. In the case of GSN applied to our setting, the results contain flickering and grid-like artifacts, which our projected noise (§ 3.3) mitigates.

4.3. Model Variations

To investigate individual components of our model, we separately evaluate variations of the layout generator and refinement network.

Layout generator. The resolution of the scene layout grid and the number of samples per ray affect the quality of the volume-rendered output I_{LR}. As shown in Table 3, higher resolution and more samples lead to the best image quality (FID computed on 32×32 pixel images for speed, compared to segmented real images with gray sky pixels). To maximize the capacity of layout generation and rendering within computational limits we opt for a 256×256 feature grid with 128 samples per ray.

Refinement network. Next, we investigate the refinement stage, which upsamples and refines the layout generator output. In our full model, the refinement network operates not only on RGB images but also on inverse-depth and sky mask

Model	Samples Per Ray	Layout Resolution	FID (I_{LR})
Low	64	32	33.66
Medium	128	32	32.02
High	128	64	22.62
Full	128	256	**16.06**

Table 3. *Variations on layout generation.* Higher feature grid resolution and more samples per ray yield the best results, bounded by computational limits. For speed, FID is computed on 5K samples rendered at 32×32.

Refinement Output	Projected Noise	FID (I_{HR})		Consistency
		C_{train}	C_{random}	
I_{HR}	✗	26.30	27.08	5.08
$I_{\text{HR}}, d_{\text{HR}}, m_{\text{HR}}$	✗	23.75	27.25	5.81
$I_{\text{HR}}, d_{\text{HR}}, m_{\text{HR}}$	✓	**21.42**	**23.39**	**3.91**

Table 4. *Variations on the refinement network.* We find refining not only the low-resolution image but also the depth and sky-mask improves image quality, but can lead to jittery results. The addition of projected noise into the upsampler results in smoother frames with lower consistency error.

(Eqn. 6), and uses projected noise for spatial consistency of texture detail (Eqn. 7). As shown in Table 4, both help to improve our model's FID and consistency error.

As shown in Fig. 7 (second row), upsampling only the RGB image I_{LR} can lead to output that is inconsistent with the generated sky mask, leading to temporally unstable gaps in the final composited image. This figure also shows the effect of our geometric regularization (Eqn. 4) in reducing unwanted transparency, especially in distant terrain.

5. Discussion and conclusion

Limitations. A few drawbacks of our model include costly volume rendering limiting the resolution of I_{LR}, imperfect 3D consistency due to image-space refinement, and imperfect or repeating geometry decoded from the scene layout features. We elaborate in the supplemental.

Conclusion. We present an unconditional world generator for unbounded synthesis of persistent 3D nature scenes. We build persistent world representation by modeling scene content with a spatially extendable layout feature grid which can be decoded via volume rendering to form a terrain image. This rendered terrain is combined with a separate sky-dome, representing infinitely far content, to synthesize novel viewpoints supporting nearby and distant camera motions. Altogether, our model enables 3D consistent image generation and view synthesis of unbounded scenes learned from single-view, unposed landscape photos.

Acknowledgements. Thanks to Andrew Liu and Richard Bowen for the fruitful discussions and helpful comments. This project was part of an internship at Google.

References

[1] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 14278–14287, 2021. 15

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2, 17

[3] Richard Strong Bowen, Huiwen Chang, Charles Herrmann, Piotr Teterwak, Ce Liu, and Ramin Zabih. OCONet: Image extrapolation by object completion. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 2307–2317, 2021. 2

[4] Tim Brooks, Janne Hellsten, Miika Aittala, Ting-Chun Wang, Timo Aila, Jaakko Lehtinen, Ming-Yu Liu, Alexei A Efros, and Tero Karras. Generating long videos of dynamic scenes. *arXiv preprint arXiv:2206.03429*, 2022. 1, 2

[5] Lucy Chai, Michael Gharbi, Eli Shechtman, Phillip Isola, and Richard Zhang. Any-resolution training for high-resolution image synthesis. In *European Conference on Computer Vision*, 2022. 5, 17

[6] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 5, 6, 13, 17

[7] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 5799–5809, 2021. 5

[8] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 4090–4100, 2019. 2

[9] Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. Scenedreamer: Unbounded 3d scene generation from 2d image collections. *arXiv preprint arXiv:2302.01330*, 2023. 3

[10] Yen-Chi Cheng, Chieh Hubert Lin, Hsin-Ying Lee, Jian Ren, Sergey Tulyakov, and Ming-Hsuan Yang. In&Out: Diverse image outpainting via GAN inversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11431–11440, 2022. 2

[11] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 7781–7790, 2019. 2

[12] Min Jin Chong, Hsin-Ying Lee, and David Forsyth. StyleGAN of All Trades: Image Manipulation with Only Pretrained StyleGAN. *arXiv preprint arXiv:2111.01619*, 2021. 4, 13, 15, 16, 17

[13] Yu Deng, Jiaolong Yang, Jianfeng Xiang, and Xin Tong. Gram: Generative radiance manifolds for 3D-aware image generation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 10673–10683, June 2022. 2

[14] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 1174–1183. PMLR, 2018. 2

[15] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W Taylor, and Joshua M Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 14304–14313, 2021. 2, 3, 5, 6, 15

[16] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Neural Information Processing Systems*, 2016. 2

[17] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. DeepView: View synthesis with learned gradient descent. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 2367–2376, 2019. 2

[18] Gereon Fox, Ayush Tewari, Mohamed Elgharib, and Christian Theobalt. StyleVideoGAN: A temporal generative model using a pretrained StyleGAN. In *Proc. British Machine Vision Conf. (BMVC)*, 2021. 2

[19] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. *arXiv preprint arXiv:2302.01133*, 2023. 3

[20] Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic VQGAN and time-sensitive transformer. *arXiv preprint arXiv:2204.03638*, 2022. 2

[21] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. StyleNeRF: A style-based 3D-aware generator for high-resolution image synthesis. *arXiv preprint arXiv:2110.08985*, 2021. 2, 5

[22] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018. 1

[23] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. Gancraft: Unsupervised 3D neural rendering of Minecraft worlds. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 14072–14082, 2021. 2, 17

[24] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *Neural Information Processing Systems*, 2020. 1

[25] James Hays and Alexei A Efros. Scene completion using millions of photographs. In *ACM Trans. Graphics (SIGGRAPH North America)*, 2007. 2

[26] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Neural Information Processing Systems*, 30, 2017. 6

[27] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 2

[28] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *Neural Information Processing Systems*, volume 31, 2018. 2

[29] Ronghang Hu, Nikhila Ravi, Alexander C. Berg, and Deepak Pathak. Worldsheet: Wrapping the world in a 3D sheet for view synthesis from a single image. In *Proc. Int. Conf. on Computer Vision (ICCV)*, 2021. 2

[30] Ali Jahanian, Lucy Chai, and Phillip Isola. On the "steerability" of generative adversarial networks. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2020. 1

[31] Varun Jampani, Huiwen Chang, Kyle Sargent, Abhishek Kar, Richard Tucker, Michael Krainin, Dominik Kaeser, William T Freeman, David Salesin, Brian Curless, et al. SLIDE: Single image 3D photography with soft layering and depth-aware inpainting. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 12518–12527, 2021. 2

[32] Wonbong Jang and Lourdes Agapito. CodeNeRF: Disentangled neural radiance fields for object categories. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 12949–12958, 2021. 2

[33] Biliana Kaneva, Josef Sivic, Antonio Torralba, Shai Avidan, and William T. Freeman. Infinite images: Creating and exploring a large photorealistic virtual space. In *Proceedings of the IEEE*, 2010. 2

[34] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Neural Information Processing Systems*, 2021. 4, 17

[35] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 8110–8119, 2020. 3, 5, 15, 16

[36] Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. Drivegan: Towards a controllable high-quality neural simulation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[37] Jing Yu Koh, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Pathdreamer: A world model for indoor navigation. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 14738–14748, 2021. 2

[38] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, Peizhao Zhang, Zijian He, Peter Vajda, Ayush Saraf, and Michael Cohen. One shot 3D photography. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 39(4), 2020. 2

[39] Wonkwang Lee, Whie Jung, Han Zhang, Ting Chen, Jing Yu Koh, Thomas Huang, Hyungsuk Yoon, Honglak Lee, and Seunghoon Hong. Revisiting hierarchical approach for persistent long-term video prediction. *arXiv preprint arXiv:2104.06697*, 2021. 2

[40] Marc Levoy and Pat Hanrahan. Light field rendering. In *ACM Trans. Graphics (SIGGRAPH North America)*, 1996. 2

[41] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. MINE: Towards continuous depth MPI with NeRF for novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12578–12588, October 2021. 2

[42] Wenbo Li, Zhe Lin, Kun Zhou, Lu Qi, Yi Wang, and Jiaya Jia. MAT: Mask-aware transformer for large hole image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10758–10768, 2022. 2

[43] Zhengqi Li, Qianqian Wang, Noah Snavely, and Angjoo Kanazawa. InfiniteNature-Zero: Learning perpetual view generation of natural scenes from single images. In *European Conference on Computer Vision*, pages 515–534. Springer, 2022. 1, 2, 5, 6, 15, 16

[44] Chieh Hubert Lin, Hsin-Ying Lee, Yen-Chi Cheng, Sergey Tulyakov, and Ming-Hsuan Yang. InfinityGAN: Towards infinite-pixel image synthesis. *arXiv preprint arXiv:2104.03963*, 2021. 2

[45] Chieh Hubert Lin, Hsin-Ying Lee, Willi Menapace, Menglei Chai, Aliaksandr Siarohin, Ming-Hsuan Yang, and Sergey Tulyakov. InfiniCity: Infinite-scale city synthesis. *arXiv preprint arXiv:2301.09637*, 2023. 3

[46] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 14458–14467, 2021. 1, 2

[47] Hongyu Liu, Ziyu Wan, Wei Huang, Yibing Song, Xintong Han, and Jing Liao. PD-GAN: Probabilistic diverse GAN for image inpainting. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 9371–9381, 2021. 2

[48] Yuchen Liu, Zhixin Shu, Yijun Li, Zhe Lin, Federico Perazzi, and Sun-Yuan Kung. Content-aware GAN compression. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 12156–12166, 2021. 2

[49] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019. 2

[50] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 359–378. Springer, 2020. 2

[51] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 3481–3490. PMLR, 2018. 15

[52] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. In *ACM Trans. Graphics (SIGGRAPH North America)*, 2019. 2

[53] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 405–421. Springer, 2020. 2, 4, 15, 17

[54] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 2

[55] Andres Munoz, Mohammadreza Zolfaghari, Max Argus, and Thomas Brox. Temporal shift GAN for large scale video generation. In *Proc. Winter Conf. on Computer Vision (WACV)*, pages 3179–3188, 2021. 2

[56] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised learning of 3D representations from natural images. In *The IEEE International Conference on Computer Vision (ICCV)*, Nov 2019. 2

[57] Michael Niemeyer and Andreas Geiger. CAMPARI: Camera-aware decomposed generative neural radiance fields. In *2021 International Conference on 3D Vision (3DV)*, pages 951–961. IEEE, 2021. 2

[58] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 11453–11464, 2021. 2

[59] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3D Ken Burns effect from a single image. *ACM Trans. Graphics*, 38(6):1–15, 2019. 2

[60] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 13503–13513, June 2022. 2, 5

[61] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3D using 2D diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 2

[62] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 5, 17

[63] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 12179–12188, 2021. 5, 13, 14

[64] Daniel Rebain, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi. LOLNeRF: Learn from one look. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1558–1567, 2022. 2

[65] Xuanchi Ren and Xiaolong Wang. Look outside the room: Synthesizing a consistent long-term 3D scene video from a single image. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 3563–3573, 2022. 2

[66] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *Proc. European Conf. on Computer Vision (ECCV)*, 2020. 2

[67] Chris Rockwell, David F Fouhey, and Justin Johnson. PixelSynth: Generating a 3D-consistent experience from a single image. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 14104–14113, 2021. 2

[68] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3D priors. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 14356–14366, 2021. 2

[69] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022. 2

[70] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[71] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: Generative radiance fields for 3D-aware image synthesis. *Neural Information Processing Systems*, 33:20154–20166, 2020. 2, 5

[72] Prafull Sharma, Ayush Tewari, Yilun Du, Sergey Zakharov, Rares Ambrus, Adrien Gaidon, William T Freeman, Fredo Durand, Joshua B Tenenbaum, and Vincent Sitzmann. Seeing 3D objects in a single image via self-supervised static-dynamic disentanglement. *arXiv preprint arXiv:2207.11232*, 2022. 3

[73] Yuan Shen, Wei-Chiu Ma, and Shenlong Wang. SGAM: Building a virtual 3D world through simultaneous generation and mapping. In *Neural Information Processing Systems*, 2022. 2

[74] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *arXiv preprint arXiv:2007.06600*, 2020. 1

[75] Lixin Shi, Haitham Hassanieh, Abe Davis, Dina Katabi, and Fredo Durand. Light field reconstruction using sparsity in the continuous fourier domain. In *ACM Trans. Graphics (SIGGRAPH North America)*, 2014. 2

[76] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3D photography using context-aware layered depth inpainting. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[77] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022. 2

[78] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 14144–14153, 2021. 5, 13

[79] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. StyleGAN-V: A continuous video generator with the price, image quality and perks of StyleGAN2. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3626–3636, 2022. 2

[80] Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. EpiGRAF: Rethinking training of 3D GANs. In *Neural Information Processing Systems*, 2022. 2, 18

[81] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov,

Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2149–2159, 2022. 2

[82] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 10521–10530, 2019. 2

[83] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[84] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3D scene inference via view synthesis. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 302–317, 2018. 2

[85] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 1526–1535, 2018. 2

[86] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017. 2

[87] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Neural Information Processing Systems*, 2016. 2

[88] Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 1020–1028, 2017. 2

[89] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 4690–4699, 2021. 2

[90] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs. In *Neural Information Processing Systems*, pages 879–888, 2017. 2

[91] Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 1399–1408, 2019. 2

[92] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 7467–7477, 2020. 2

[93] Zongxin Yang, Jian Dong, Ping Liu, Yi Yang, and Shuicheng Yan. Very long natural scenery image prediction by outpainting. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 10561–10570, 2019. 2

[94] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. In *Proc. Int. Conf. on Computer Vision (ICCV)*, 2019. 2

[95] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. PixelNeRF: Neural radiance fields from one or few images. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 4578–4587, 2021. 2, 3

[96] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 5505–5514, 2018. 2

[97] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 4471–4480, 2019. 2

[98] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks. In *The Tenth International Conference on Learning Representations*, 2022. 2

[99] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2021. 2

[100] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *ACM Trans. Graphics (SIGGRAPH North America)*, 2018. 2

In supplemental materials, we investigate an alternative 3D feature representation based on extendable triplane units A. We provide additional implementation details of our method in Section B, additional ablations in Section C, and we provide further discussion on our model in Section D.

A. Extended Triplane Variation

Instead of decoding the scene from a 2D layout feature grid and height of a 3D point above this layout plane, we also experiment with a model variation that adds vertical support planes parallel to the XY and YZ planes. Thus, the layout features are described by a 2D extended XZ layout feature grid, and sets of orthogonal support planes shown in pink in Fig. 8-left. Decoding a given 3D point projects the point to the XZ plane, the four nearest vertical planes (two parallel to XY and two parallel to YZ, which are weighted linearly according to the distance of the point from each plane).

Qualitatively, the triplane model achieves more geometry diversity, with more mountainous terrain compared to the feature layout model. We attribute this to the additional support provided from the vertical feature planes. Additionally, the vertical feature planes allow for a lighter decoding network with higher neural rendering resolution, allowing for faster video rendering and improved temporal consistency (lower one-step consistency error) due to less reliance on a 2D upsampling operation. We show qualitative examples in Fig. 9 with video results on our project page, and quantitative evaluations in Tab. 5. Quantitatively, while this extended triplane variation does not output perform the layout model in terms of FID, we hypothesize that the FID may be impacted by two possible factors: first, this model requires inference-time camera height adjustment to avoid intersecting with increased complexity of the generated geometry, and second, interpolation between vertical feature planes qualitatively produces more muted colors compared to the real image distribution.

We also investigate the impact of using a projected 3D noise pattern as input into the extended triplane upsampler, with results in Tab. 6. While this improves FID and consistency in the layout representation, we find that the benefits of the projected noise are more limited in the extended triplane setting. Adding projected noise offers improvements in FID, but also a small increase in consistency error. Qualitatively, the model outputs are similar with and without the projected noise, perhaps attributed to decreased reliance on the upsampling operation.

B. Additional Methodological Details

B.1. Preprocessing

Dataset Filtering. To remove images in the LHQ [78] dataset that contain occluding objects close to the camera,

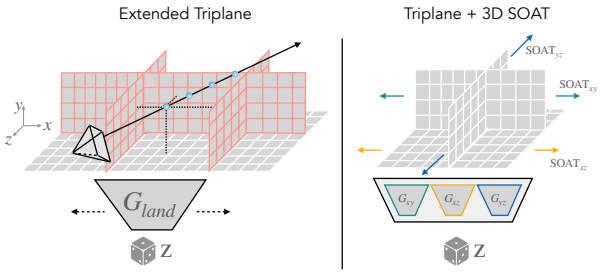


Figure 8. *Diagram of Extended Triplane Representation.* The extended triplane representation adds a sequence of orthogonal vertical feature planes outlined in pink in addition to the ground plane features outlined in white **(left)**. Each unit consists of a triplane representation [6] generated from three independent generators – G_{XY}, G_{XZ}, and G_{YZ} – tied to the same latent code and mapping network **(right)**. At inference time, the features of each generator are stitched along the appropriate dimensions using the SOAT procedure [12].

Model	FID			Consistency	Render Time (s)
	C_{train}	C_{forward}	C_{random}		
Extended Layout	**21.42**	**26.67**	**23.39**	3.56	8.49
Extended Triplane	24.47	34.89	34.76	**2.29**	**0.16**

Table 5. *Extended Layout vs. Extended Triplane* While the extended layout representation presented in the main paper attains better image quality (lower FID scores), the extended triplane representation offers improved consistency (lower one-step consistency error) and dramatically faster video rendering (as the layout model requires supersampling for video smoothness). We hypothesize that inference-time camera adjustments and interpolation between vertical feature planes may negatively impact FID for the extended triplane model, despite its ability to generate more complex and diverse landscape geometry.

Model	FID			Consistency
	C_{train}	C_{forward}	C_{random}	
Without Noise	**24.47**	34.89	34.76	**2.29**
With 3D noise	25.31	**33.30**	**33.28**	3.06

Table 6. *Effect of 3D Projected Noise* Adding projected noise into the upsampler of the extendable triplane representation offers improvements in FID but is slightly more inconsistent, but still more consistent than the layout model.

we apply filtering criteria to construct the training dataset. Using the segmentation output of DPT [63], we detect the sky region and boundaries of the resulting binary sky mask. As the segmentation results can include small regions with inconsistent labels (*e.g.* small holes in the sky), we remove all bounded regions with area under 250 pixels to create a more unified sky mask. Next, using this segmentation mask we filter out images for which any of the following hold: (1) there are more than three bounded sky regions, (2) more

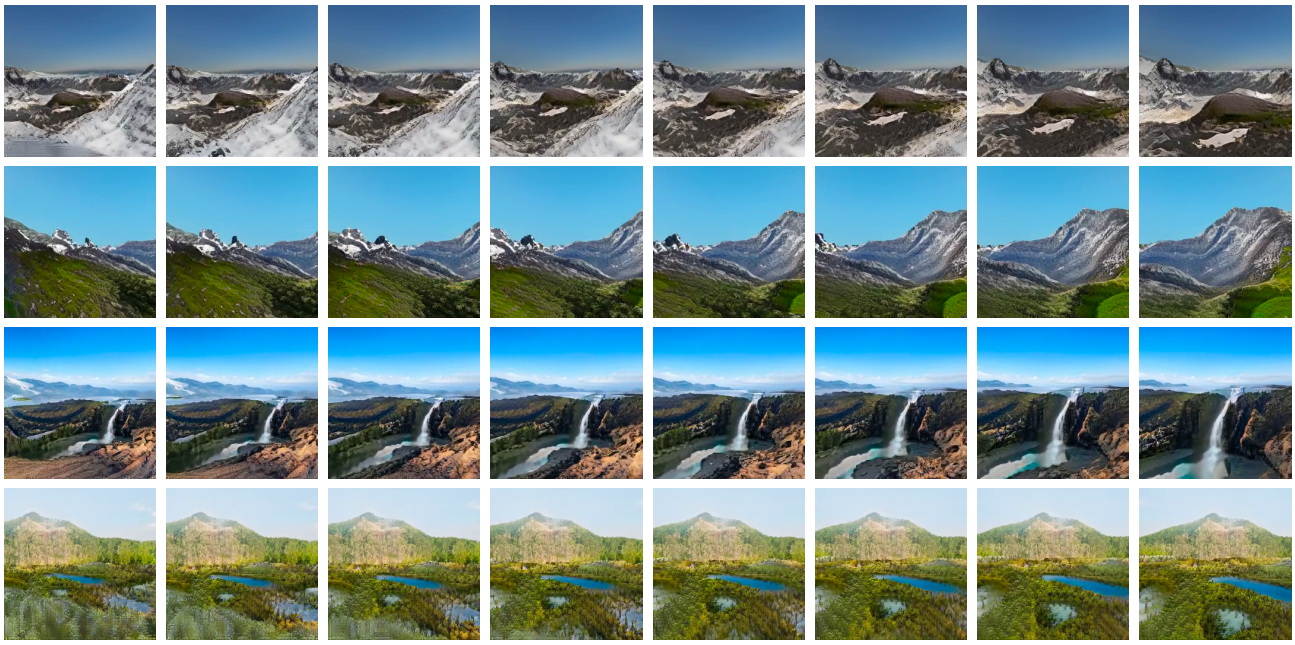
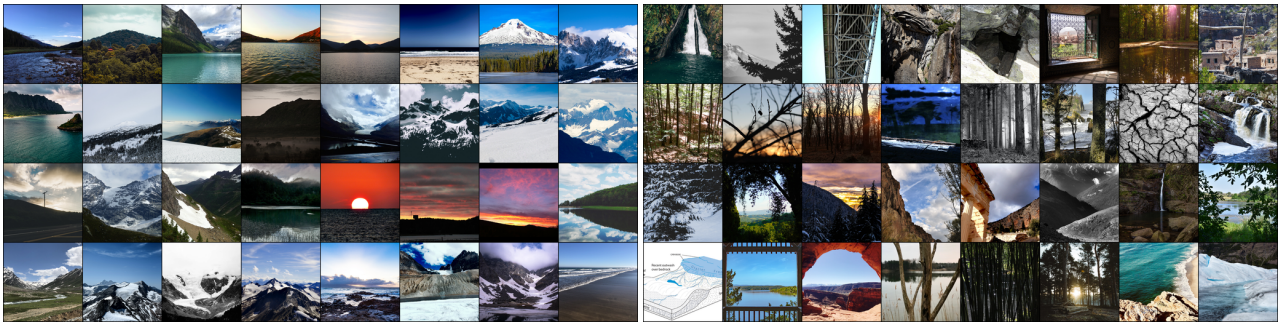


Figure 9. *Extendable Triplane Visualization.* Qualitative examples of rendering from the extendable triplane representation. This representation results in larger scene and geometry diversity compared to the layout feature representation, with improved 3D consistency.



(a) Training Images (b) Removed Images

Figure 10. *Result of dataset filtering.* The dataset filtering step (a) retains images that contain sufficient sky pixels near the top of the image, and (b) removes images that are not typical images of landscapes. These atypical images include images without sky pixels, or images with nearby occluding objects such as windows or trees. The filtering criteria is based on sky segmentation and disparity estimation obtained from DPT [63].

than 90% of the scene is not sky pixels, (3) more than 40% of the upper one-fifth of the image is not sky pixels, and (4) less than 80% of the lower quarter of the image is not sky pixels. The first three criteria are meant to filter out images that contain occluding structures (such as trees or windows) or images in which there is no sky region present. The fourth criteria is meant to filter out images taken from unusual camera angles (such as from underneath a bridge). Using the monocular depth prediction from DPT, we also remove images containing too many vertical edges: images are removed if the 99th percentile of the pixel-wise finite difference is greater than 0.05, which tends to be indicative of trees or man-made buildings. Fig. 10 shows examples of images that were retained for training, and those that were filtered out.

Disparity Normalization. Using the monocular depth prediction from DPT, we normalize the disparity values between 0 and 1 using the 1st and 99th percentile values per image. Next, we clip the minimum disparity for non-sky regions and rescale the disparity values to correspond to the near and far bounds used in volumetric rendering (see § B.2.2). We use 0.05 for our clip value and 1/16 for the scale factor; this means that after normalization, the disparity values for non-sky pixels range from 1/16 to 1. The disparity for the

sky pixels is clamped at zero.

Camera Poses. We sample training camera poses with a random (x, z) position within the layout grid, and a rotation such that the near half of the view frustrum lies entirely within the training grid. To simulate the forward motion of InfiniteNature-Zero [43], we move the camera forward a distance equivalent to 100 steps of InfiniteNature-Zero, corresponding to roughly half of the scene layout grid. To evaluate view extrapolation, we randomize the position and rotation of the cameras at inference time. These settings are illustrated in Fig. 11.

B.2. Training and Implementation

B.2.1 Training objective

Each stage of our model is trained following the StyleGAN2 objective [35], with a non-saturating GAN loss V and R_1 regularization [51]:

$$\begin{aligned} &V(D, G(\mathbf{z}), I) = D(I) - D(G(\mathbf{z})), \\ &R_1(D, I) = ||\nabla D(x)||^2, \\ &G = \arg\min_G \max_D \; \mathbb{E}_{\mathbf{z}, I\sim\mathcal{D}}\, V(D, G(\mathbf{z}), I) + \\ &\qquad \frac{\lambda_{R_1}}{2} R_1(D, I), \end{aligned} \tag{10}$$

where G, D refer to the corresponding generator and discriminator networks at each training stage, and x refers to real images sampled from dataset $\mathcal{D}$. Additional auxiliary losses for each part of the model are described in the following sections.

B.2.2 Layout Generator

Our layout generator is based on the architecture from GSN [15], which is comprised of two components: G_{land}, which synthesizes the scene layout grid, and M which decodes the 2D layout feature into a 3D feature.

The layout generator G_{land} follows StyleGAN2 [35], which generates a 256×256 grid of features $f_{\text{land}} \in \mathbb{R}^{32}$. G_{land} contains three mapping layers and the maximum channel dimension is capped at 256; all other parameters are unchanged from StyleGAN2.

The network M is modeled after the style-modulated MLP from CIPS [1], containing eight layers with a hidden channel dimension of 256 and producing features $f_{\text{color}} \in \mathbb{R}^{128}$. The constant input to M is replaced with the y-coordinate (height above the ground plane), and the modulation input is the interpolated feature from f_{land}.

We adapt the rendering procedure of GSN to handle unbounded outdoor scenes. For volumetric rendering, we set the near bound to 1 and the far bound to 16, which corresponds to the scale factor used in disparity normalization during data preprocessing. Each scene layout feature has a unit width of 0.15, such that the full width of the feature grid is $256 \times 0.15 = 38.4$, which is slightly over twice the far bound distance. We omit positional encoding from M, as we found that including positional encoding yielded grid-aligned artifacts in generated images; we also omit the view direction input. Camera rays are sampled using $\text{FOV} = 60°$ with linearly spaced sampling between the near bound and the far bound. We use inverse-depth (disparity) supervision rather than depth supervision so that we can represent content at infinite distances. This also encourages the terrain generator to create empty space in the sky content, which will be filled with the skydome generator.

We use the volumetric rendering equations from NeRF [53], in which the weights w_i of the i-th point along a ray depends on densities σ which is predicted by multi-layer perceptron M and the distance between samples δ:

$$\alpha_i = 1 - \exp\left(-\sigma_i \delta_i\right), \quad w_i = \alpha_i \exp\Big(-\sum_{j=1}^{i-1} \sigma_j \delta_j\Big). \tag{11}$$

Our training procedure for the layout decoder follows that of GSN [15], which provides the real RGB image I_{RGB} and disparity d (obtained from DPT) to the discriminator $I = \{I_{\text{RGB}}, d\}$, and also adds a reconstruction loss on real images using a decoder network $G\phi$ on discriminator features $D\phi$:

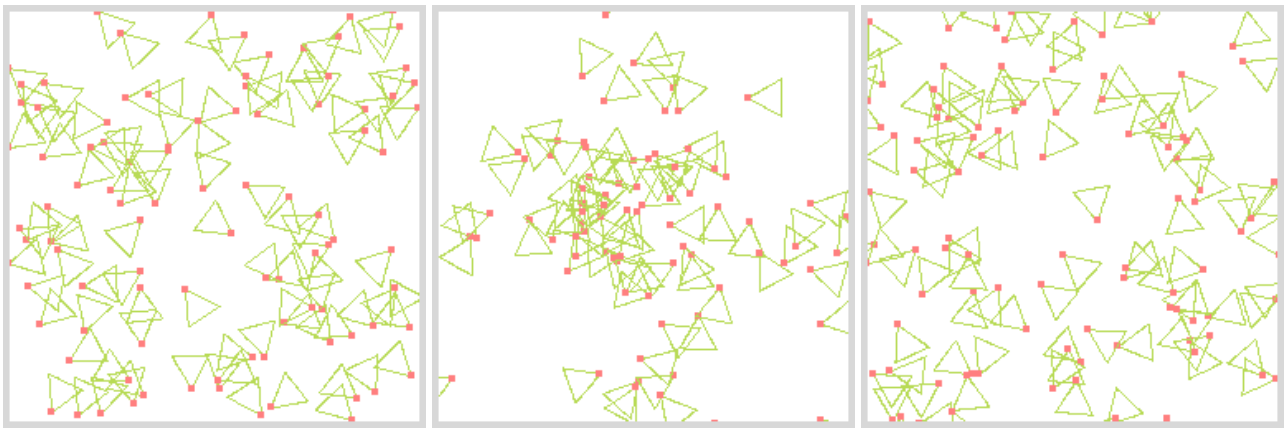
$$\mathcal{L}_{\text{rec}} = (I - G\phi(D\phi(I)))^2. \tag{12}$$

The full GAN objective follows Eqn. 10 with weights $\lambda_{R_1} = 0.01$ and $\lambda_{\text{rec}} = 1000$, and we follow the optimizer settings from StyleGAN2 and train for 12M image samples.

Because the layout decoder tends to generate semi-transparent geometry, which also causes unrealistic sky masks, we regularize the geometry following Eqn. 4, and add the sky mask into the discriminator. We finetune with this additional loss for 400k samples with $\lambda_{\text{transparent}}$ which linearly increases from zero to $\lambda_{\text{transparent}} = 80$ over the finetuning procedure.

B.2.3 Layout Extension

We use the procedure of SOAT [12] in two dimensions to smoothly transition between adjacent feature grids sampled from independent latent codes. SOAT proceeds by operating on 2x2 sub-grids and stitching each layer of intermediate features in the generator (Fig. 12). To start, we simply concatenate the StyleGAN constant tensors, to obtain a feature grid f_0 of size $2H_0 \times 2W_0$, where H_0 and W_0 are the height and width of the constant tensor. For each subsequent layer f_{l+1}, we modulate the weights G_l with each of four corner latent codes (after applying the mapping network to obtain the style-code) and apply it in a fully convolutional manner to f_l, obtaining $f_{k,l+1}$ of size $2H_l \times 2W_l$. Then, we multiply each of $f_{k,l+1}$ with bilinear interpolation weights β and take



(a) Training camera distribution (b) Cameras after forward motion (c) Random cameras

Figure 11. *Illustration of camera distributions.* (a) Cameras used for training are sampled with a random translation uniformly over the scene layout feature grid, with rotation sampled to overlap with this feature grid. To evaluate view extrapolation, we (b) move the cameras forward a distance equivalent to 100 steps of InfiniteNature-Zero [43], corresponding to roughly halfway across the scene layout grid, or (c) randomly sample a random translation and random rotation.

the sum to obtain f_{l+1}. This procedure is repeated for each layer of the generator, obtaining a an output feature grid of size $2H \times 2W$. To reduce the effect of padding, these output feature grids are tiled in an overlapping manner, with a 25% overlap on each side and with weights that linearly decay to zero away from the center of the tile.

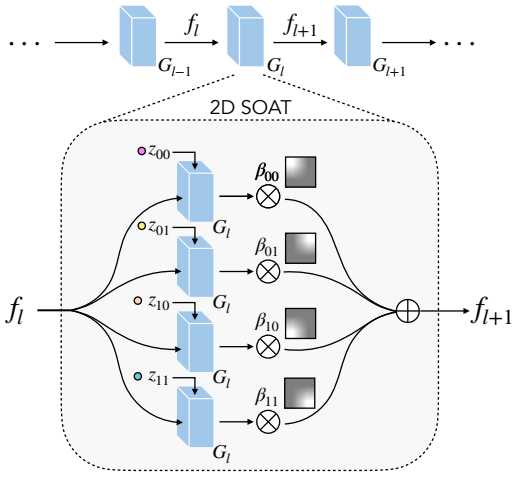


Figure 12. *Layout Extension.* We adapt the procedure in SOAT [12] for 2D layout extension. Operating on each layer of the generator, we take the incoming feature grid f_l, and construct the outgoing feature grid using the generator weights conditioned on each corner latent code $\mathbf{z}$ (the conditioning uses weight modulation on the mapping network outputs in StyleGAN2 [35]). Then, these four outgoing feature maps are multiplied with bilinear weights β and the result is summed, to obtain the blended feature for the next layer f_{l+1}.

B.2.4 Refinement Network

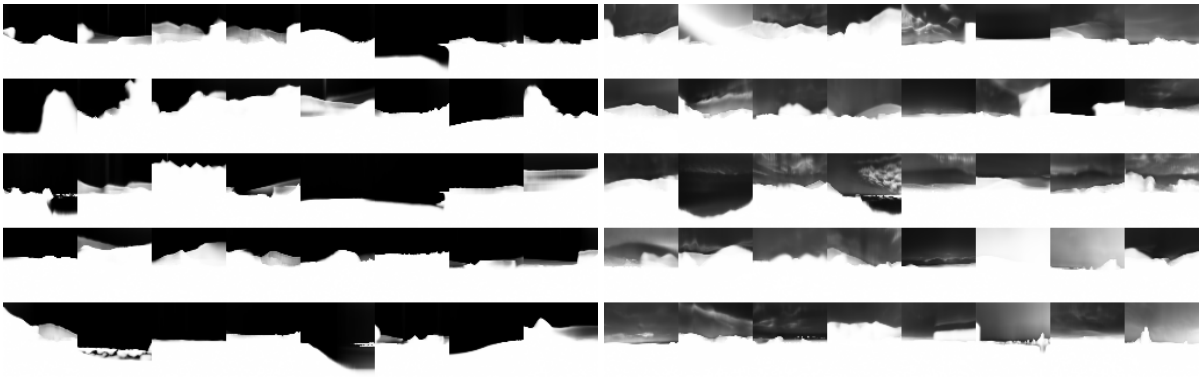
The refinement network G_{up} uses a truncated StyleGAN2 backbone, which replaces the feature input of the 32×32 block with the 32×32 rendered feature f_{im} and initial image I_{LR}, depth d_{LR}, and sky mask m_{LR}. The skip connection of the upsampler takes in I_{LR}, d_{LR}, m_{LR} and predicts I_{HR}, d_{HR}, and m_{HR}. Following the noise injection operation in StyleGAN2, we replace the image-space 2D noise tensor with our 3D-consistent projected noise (Eqn. 7). This network uses two mapping layers, taking as input the style latent vector from G_{land}.

We add an additional objective to encourage consistency between the refined color pixels and the sky mask:

$$\begin{aligned} \mathcal{L}_{\text{consistency}} &= |d_{\text{HR}} - d_{\text{LR}\uparrow}| + |m_{\text{HR}} - m_{\text{LR}\uparrow}|, \\ \mathcal{L}_{\text{sky}} &= \exp(-20 * \sum_{c} |I_{\text{HR}}[c]|) * m_{\text{HR}}. \end{aligned} \tag{13}$$

The loss $\mathcal{L}_{\text{consistency}}$ encourages the high resolution depth and mask outputs to match their upsampled low resolution counterparts (this results in a smoother outcome compared to downsampling the high resolution outputs). The loss $\mathcal{L}_{\text{sky}}$ encourages pixel colors to be nonzero (reserved for the gray sky color) when $m_{\text{HR}} = 1$, by summing over the three channels c of the predicted image I_{HR}; this is meant to encourage the RGB colors produced refinement network to be consistent with the mask and depth outputs. The refinement network is trained with the GAN objective (Eqn. 10) with weights $\lambda_{R1} = 4$, $\lambda_{\text{consistency}} = 5$, and $\lambda_{\text{sky}} = 100$, and the discriminator loss is applied only on the RGB images.

Due to the computational costs of volume rendering, we train the refinement network on 32×32 inputs to produce 256×256 outputs. For 30 fps video visualizations, we su-



(a) Accumulated ray density with separate skydome

(b) Accumulated ray density without separate skydome

Figure 13. *Training without a separate skydome.* We supervise the sky content with zero inverse-depth (infinite distance) to ensure that the camera does not intersect the sky as the layout features are extended. As such, we model content at infinite distances with a separate skydome model, such that the terrain model treats sky regions as empty space (left). Without the separated skydome, the model is forced to put sky content at finite distances leading to foggy, semi-transparent content near the camera (right).

persample the camera rays at 8x spatial density and apply depth-based filtering to the noise input to improve video smoothness; however all metrics in the paper are computed without supersampling for additional smoothness.

We note that while StyleGAN3 [34] is intended to resolve the texture sticking effect caused by the noise input in StyleGAN2, replacing G_{up} with a StyleGAN3 backbone resulted in worse image quality in our setting with FID 67.90, compared to FID 21.42 for our final model.

B.2.5 Skydome Generator

The skydome generator takes as input the CLIP [62] embedding of a single terrain image, and predicts a sky output that is consistent with the terrain. The generator architecture follows StyleGAN3 [34] adapted with cylindrical coordinates to generate 360° panoramas [5].

For the terrain input, we take the filtered LHQ dataset and select the non-sky pixels with normalized disparity greater than 1/16 (this leaves some background mountains to be predicted). We follow the training procedure from [5] with a few adaptations. In addition to concatenating the CLIP embedding of the terrain image to the style-code, the generated sky is composited with the terrain input prior to the discriminator with 50% probability, which is compared to full RGB images from LHQ. The 50% compositing behavior ensures that the bottom of the generated skydome can still appear realistic (when unmasked), while also matching provided terrain image (when masked). This portion is trained with the $\lambda_{R1} = 2$ in the GAN objective (Eqn. 10), with randomly sampled cylindrical coordinates and a cross-frame discriminator applied to the boundary of two adjacent frames.

B.3. Extendable Triplane Implementation

To construct the extendable triplane representation, we modify the triplane model from EG3D [6] to generate three planes from independent synthesis networks G_{XY}, G_{XZ}, and G_{YZ}, tied to the same latent code and mapping network. Similar to our layout feature model, we train the terrain generator on sky-segmented images and disparity maps as input into the low-resolution discriminator to help the model learn geometry. The upsampler portion of this model and the training procedure is the same as EG3D, using $\lambda_{R1} = 10$. To prevent the model from rendering the segmented sky color (we use white for the sky color, following the background color of NeRF [53]), we finetune the model penalizing for white pixels when the sky mask is one:

$$\mathcal{L}_{\text{sky}} = \exp(-5 * \sum_{c} (I_{\text{LR}}[c] - 1) * m_{\text{LR}}. \qquad (14)$$

The finetuning operation is performed for 400K samples with λ_{sky} increasing from zero to 40 during training. At inference time, we perform SOAT [12] feature stitching to each generator along the appropriate dimensions to obtain the extended triplane representation. As the skydome model does not train on generated images, we use the same skydome model as before. We use 50 randomly sampled camera poses for training, which improves the geometry diversity (more mountainous terrain) the compared to using 1K random training poses.

C. Additional Experiments

C.1. Training without a separate skydome

Modelling faraway content separately is a common strategy in unbounded scene-reconstruction [2, 23]. To ensure that we cannot intersect the skydome as we arbitrarily extend the layout features, we use zero inverse-depth for sky pixels, which can only render a solid color as the weight of all points

along the ray must be zero to obtain zero inverse-depth. In this experiment, we train I_{LR} using the same training strategy as our final I_{LR} model, but instead supervise with full RGB images rather than sky-segmented RGB images. This corresponds to training the model without a separate skydome. We find that without the separate skydome, the model learns incorrect geometry, as it is forced to place some density at finite distances in order to render content in the sky to match the training distribution. Figure 13 shows the difference in ray accumulations from models trained with the skydome (prior to opacity regularization) and without the skydome. The model without the skydome places semi-transparent content in the sky region, which creates a fog-like effect when moving the camera throughout the landscape.

C.2. Changing the number of sampled cameras

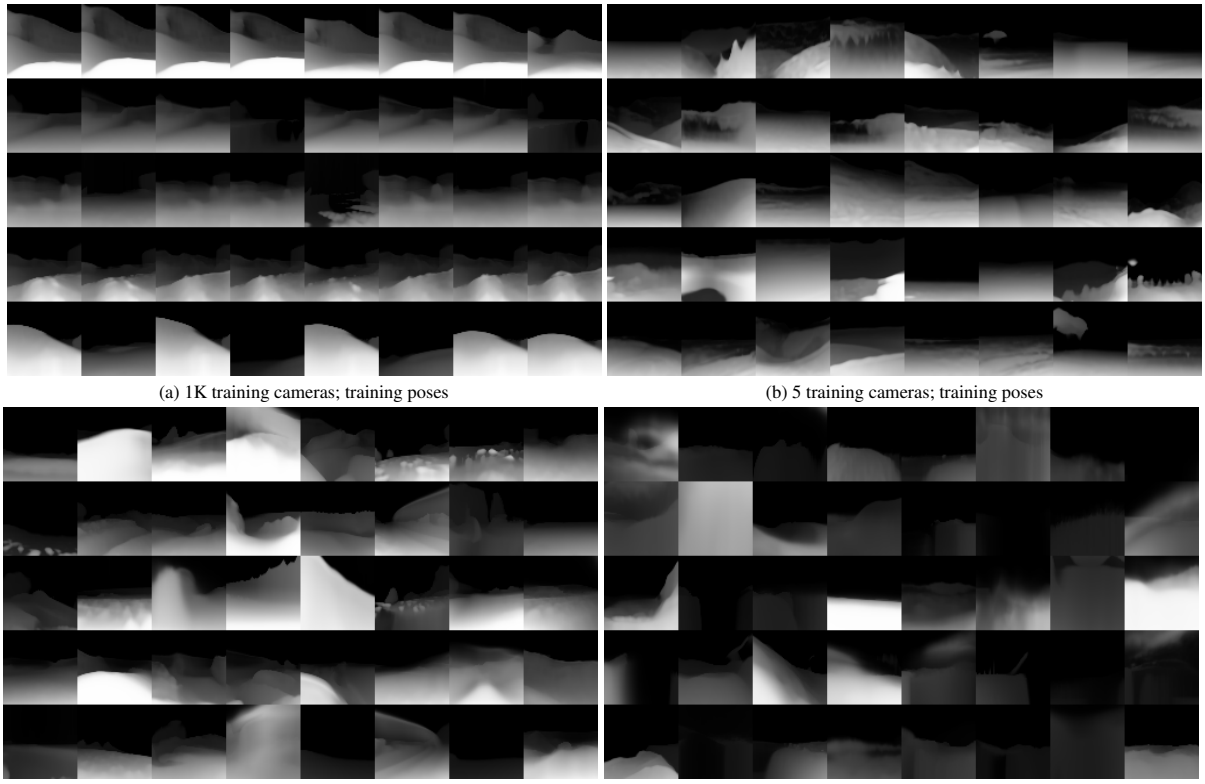
We train our model using a set of one thousand cameras with randomly sampled translations within the layout feature grid, and rotations such that the camera view frustum overlaps with the feature grid. However, one limitation of this training strategy is that we find the model can learn repeating geometry, such that the rendered disparity map may look similar when sampling different random latent codes at the same camera position, despite the pixel color values being different. We hypothesize that the diversity of camera poses sampled during training may obscure the repeating geometry effect from the discriminator, as images sampled from different camera poses will appear different in terms of both color and geometry.

To investigate this effect, we train another model using only five camera poses during training. The disparity maps per camera pose show more diversity in this setting, however we find that this setting results in "holes" and incorrect geometry in the landscape when moving the camera away from the training poses, illustrated in Figure 14. We use one thousand training cameras as our default setting, but a more optimal setting may involve fewer training cameras, while still ensuring adequate coverage over the feature grid.

D. Discussion

A limiting factor of our method is the reliance on a volume rendering operation to decode the 2D layout feature grid into a 3D feature at each sampled point along the ray. Due to this operation, the rendered output I_{LR} can only be trained at low resolution (32x32), and does not learn to generate detailed textures. (In contrast to NeRF-style models which can use per-ray supervision, we must render a complete image as an input for the discriminator.) We rely on a refinement module to upsample the result and add additional textures, but any refinement in image space is prone to losing 3D consistency. Our extended triplane variation reduces the computational expense of volume rendering by reducing the capacity of the decoder MLP and increasing the capacity of the feature representation, thus allowing for neural rendering at 64x64 resolution (we find that geometry degrades at higher resolutions) and decreasing reliance on the upsampler. While we did not find improvements when training on rendered patches, improved patch sampling techniques could help in adding more detail to the rendered result [80].

As our model does not have explicit 3D or aerial supervision, we find that it may generate unnatural or repeating geometry. This can appear as thin mountains, sloping water, or hills of a similar shape but different appearance when sampling from different random noise codes.



(a) 1K training cameras; training poses

(b) 5 training cameras; training poses

(c) 1K training cameras; independent test poses

(d) 5 training cameras; independent test poses

Figure 14. *Adjusting the set of training cameras.* We plot disparity maps corresponding to training with one thousand cameras, and five cameras. (a) With our default setting of one thousand training cameras with camera origins uniformly sampled over the layout feature grid, we find that the model can learn repeating geometry, such that the disparity map generated from the same pose but different latent codes tends to look similar (each row corresponds to the same pose), despite the RGB colors appearing different. (b) With fewer training cameras, the models learns more diversity in the rendered geometry, where again each row corresponds to the same camera pose. (c & d) However, the model trained with one thousand cameras generalizes better to an independent set of cameras, whereas the model trained with five cameras has a greater frequency to put holes in the decoded landscape (evidenced by completely black disparity maps, or disparity maps that have no nearby content and thus are darker overall) or regions of solid content without sky (evidenced by disparity maps that do not fade to black near the top of each image). We use one thousand training cameras as our default setting, but a more optimal setting may involve fewer training cameras, while still ensuring adequate coverage over the feature grid.