Persistent Nature: A Generative Model of Unbounded 3D Worlds Supplemental Material

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. 1-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. 2 with video results on our project page, and quantitative evaluations in Tab. 1. 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. 2. 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.

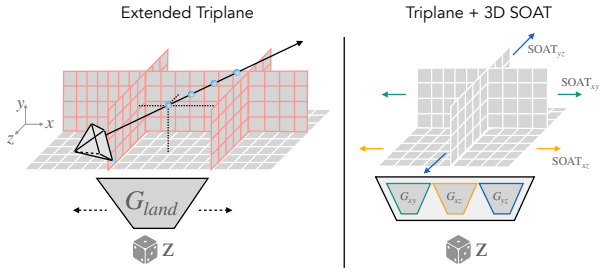


Figure 1. *Diagram of Extended Triplane Representation.* The extended triplane representation adds a sequence of orthogonal vertical feature planes outlined in pink in addition to the ground plane features outlined in white **(left)**. Each unit consists of a triplane representation [4] generated from 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 [5].

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 1. *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.

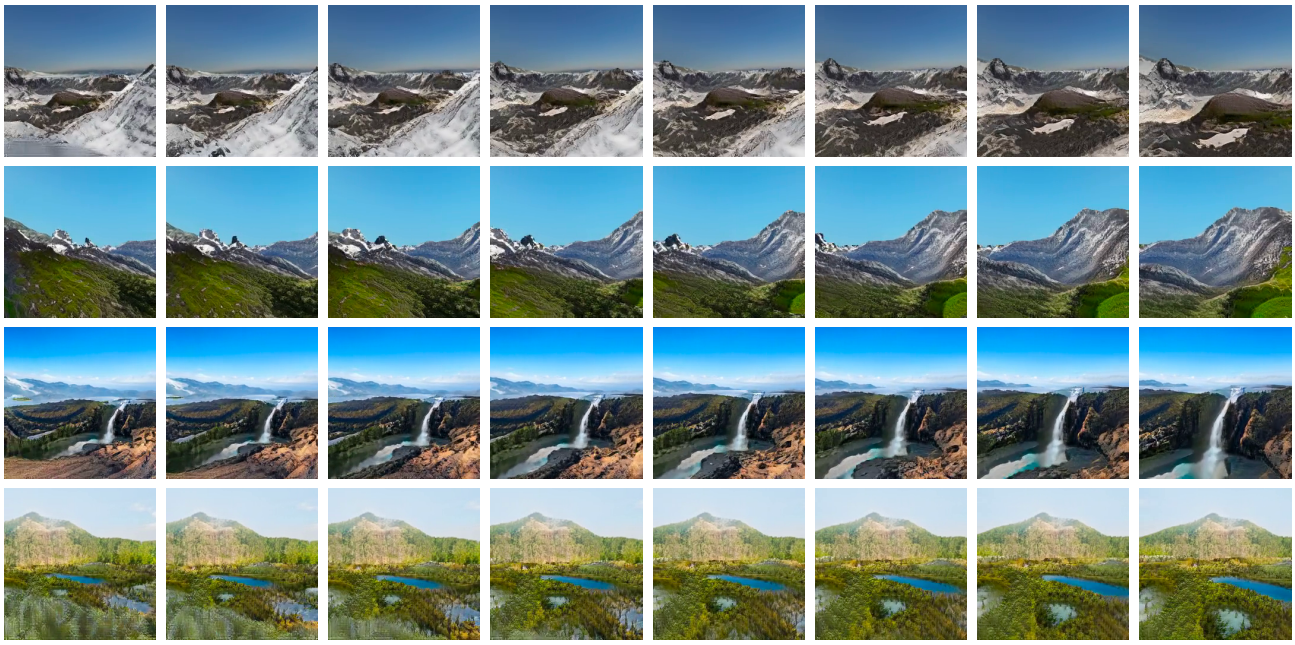


Figure 2. *Extendable Triplane Visualization.* Qualitative examples of rendering from the extendable triplane representation. This representation results in larger scene and geometry diversity compared to the layout feature representation, with improved 3D consistency.

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 2. *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.

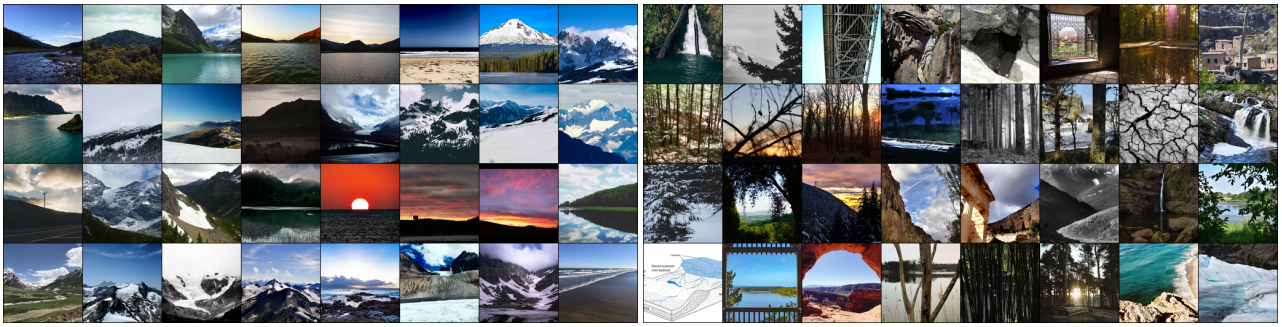
B. Additional Methodological Details

B.1. Preprocessing

Dataset Filtering. To remove images in the LHQ [15] dataset that contain occluding objects close to the camera, we apply filtering criteria to construct the training dataset. Using the segmentation output of DPT [14], 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 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. 3 shows examples of images that were retained for training, and those that were filtered out.

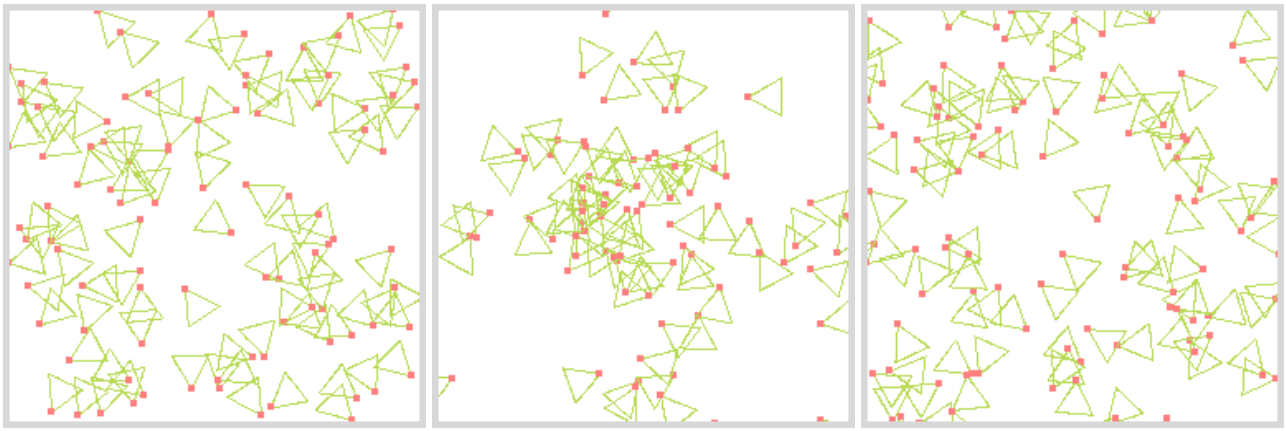
Disparity Normalization. Using the monocular depth prediction from DPT, we normalize the disparity values between 0 and 1 using the 1st and 99th percentile values per image. Next, we clip the minimum disparity for non-sky regions and rescale the disparity values to correspond to the near and far bounds 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 [10], we move the camera forward a distance equivalent to 100 steps of InfiniteNature-Zero, corresponding to roughly half of the scene layout grid. To



(a) Training Images (b) Removed Images

Figure 3. *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 [14].



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

Figure 4. *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 [10], corresponding to roughly halfway across the scene layout grid, or (c) randomly sample a random translation and random rotation.

evaluate view extrapolation, we randomize the position and rotation of the cameras at inference time. These settings are illustrated in Fig. 4.

B.2. Training and Implementation

B.2.1 Training objective

Each stage of our model is trained following the StyleGAN2 objective [9], with a non-saturating GAN loss V and R_1 regularization [11]:

$$
\begin{aligned}
&V(D, G(\mathbf{z}), I) = D(I) - D(G(\mathbf{z})), \\
&R_1(D, I) = ||\nabla D(x)||^2, \\
&G = \arg\min_G \max_D \; \mathbb{E}_{\mathbf{z}, I\sim\mathcal{D}} \; V(D, G(\mathbf{z}), I) + \\
&\quad \frac{\lambda_{R_1}}{2} R_1(D, I),
\end{aligned} \tag{1}
$$

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 [6], 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 [9], 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 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 [12], 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:

$$\alpha_i = 1 - \exp\left(-\sigma_i \delta_i\right), \quad w_i = \alpha_i \exp\Big(-\sum_{j=1}^{i-1} \sigma_j \delta_j\Big). \quad (2)$$

Our training procedure for the layout decoder follows that of GSN [6], 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 $D\phi$:

$$\mathcal{L}_{\text{rec}} = (I - G\phi(D\phi(I)))^2. \quad (3)$$

The full GAN objective follows Eqn. 1 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 in the main document, 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 [5] 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. 5). 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 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 50% overlap on each side and with weights that linearly decay to zero away from the center of the tile.

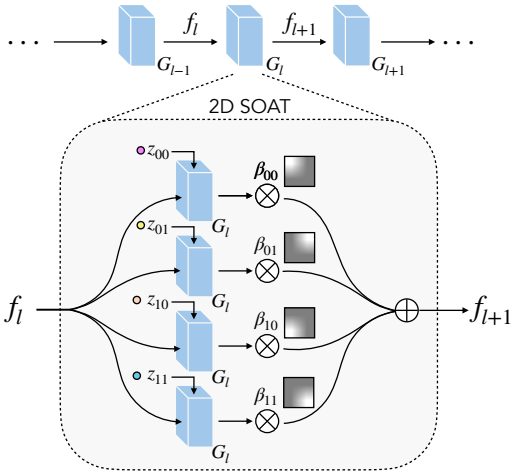
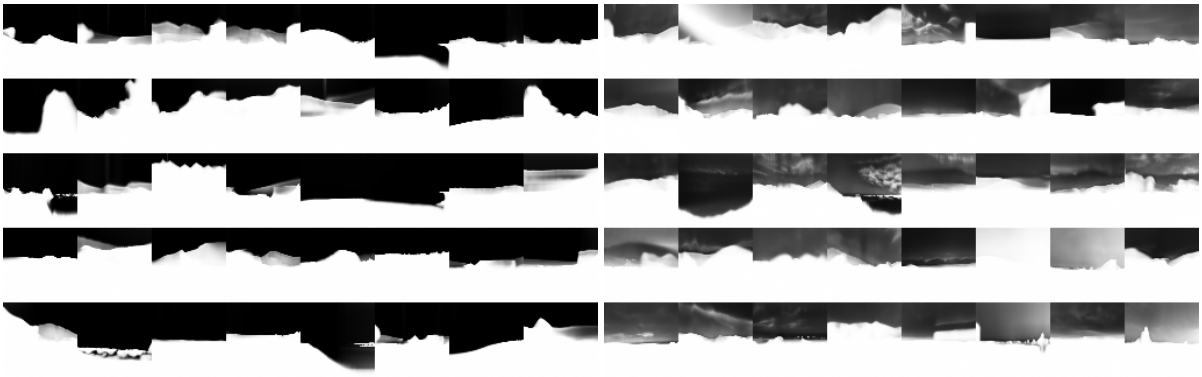


Figure 5. *Layout Extension.* We adapt the procedure in SOAT [5] for 2D layout extension. Operating on each layer of the generator, we take the incoming feature grid f_l, and construct the outgoing feature grid using the generator weights conditioned on each corner latent code $\mathbf{z}$ (the conditioning uses weight modulation on the mapping network outputs in StyleGAN2 [9]). 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 in the main



(a) Accumulated ray density with separate skydome

(b) Accumulated ray density without separate skydome

Figure 6. *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).

document). 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:

$$\begin{aligned} \mathcal{L}_{\text{consistency}} &= |d_{\text{HR}} - d_{\text{LR}\uparrow}| + |m_{\text{HR}} - m_{\text{LR}\uparrow}|, \\ \mathcal{L}_{\text{sky}} &= \exp(-20 * \sum_{c} |I_{\text{HR}}[c]|) * m_{\text{HR}}. \end{aligned} \tag{4}$$

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. 1) 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 supersample 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 [8] 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 [13] embedding of a single terrain image, and predicts a sky output that is consistent with the terrain. The generator architecture follows StyleGAN3 [8] adapted with cylindrical coordinates to generate 360° panoramas [3].

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 [3] 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. 1), 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 [4] 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 [12]), we finetune the model penalizing for

white pixels when the sky mask is one:

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

The finetuning operation is performed for 400K samples with λ_{sky} increasing from zero to 40 during training. At inference time, we perform SOAT [5] 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, 7]. 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 6 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

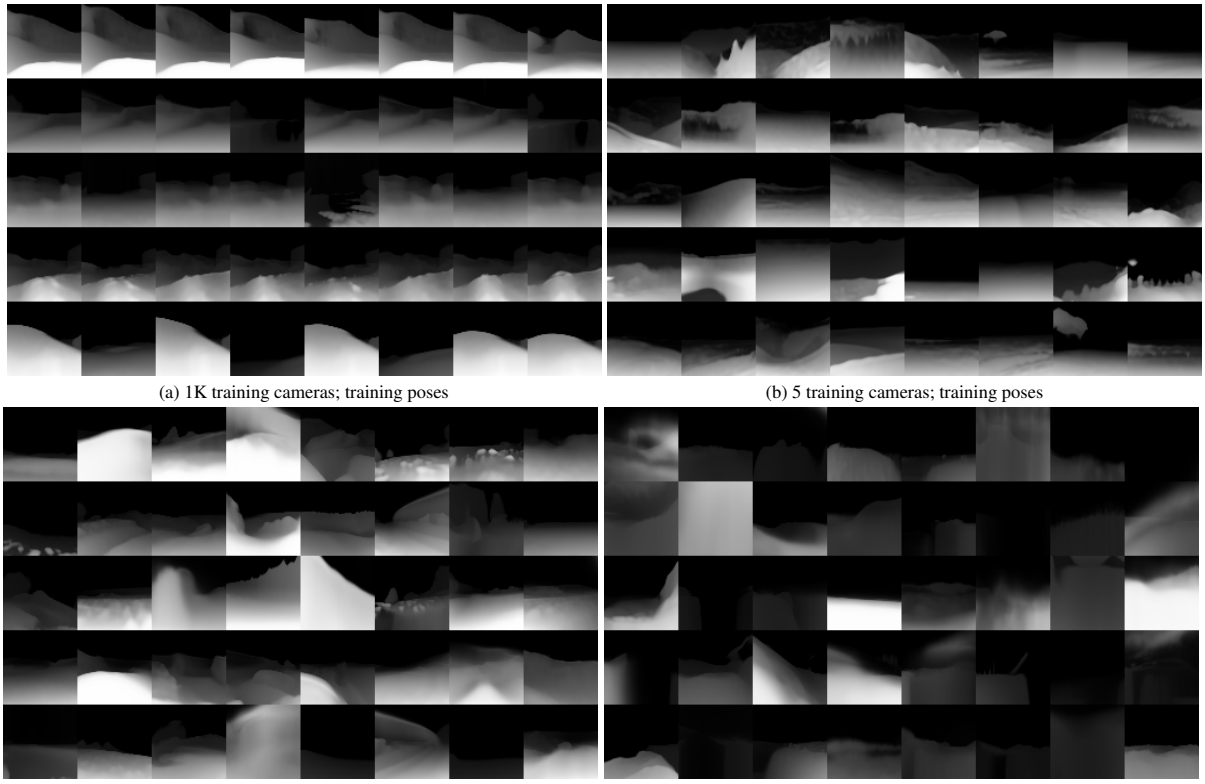
We train our model using a set of one thousand cameras with randomly sampled translations within the layout feature grid, and rotations such that the camera view frustum overlaps with the feature grid. However, one limitation of this training strategy is that we find the model can learn repeating geometry, 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 7. 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 [16].

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 7. *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.

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. 4

[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. 6

[3] 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

[4] 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, 5

[5] 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. 1, 4, 6

[6] 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. 3, 4

[7] 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. 6

[8] 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. 5

[9] 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, 4

[10] 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. 2, 3

[11] 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. 3

[12] 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. 4, 5

[13] 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

[14] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 12179–12188, 2021. 2, 3

[15] 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. 2

[16] Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. EpiGRAF: Rethinking training of 3D GANs. In *Neural Information Processing Systems*, 2022. 6