NuiScene: Exploring Efficient Generation of Unbounded Outdoor Scenes

Supplementary Material

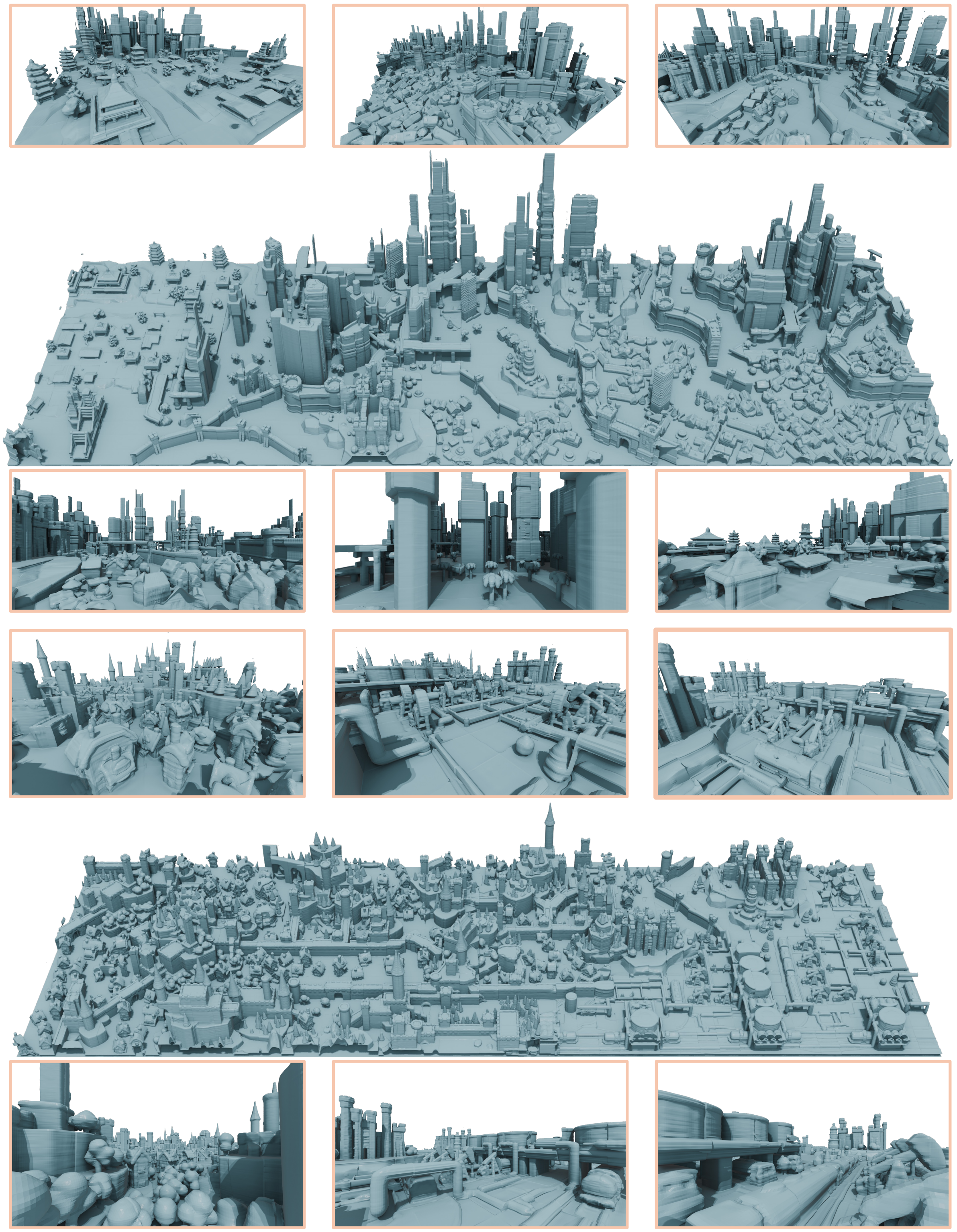


Figure 1. Additional generation results for our vector set diffusion model trained on 13 scenes. We show zoomed-in aerial shots on the top with street level views on the bottom respectively for the two scenes. The scenes shown here are of size 16×46.

A. NuiScene43 Dataset

We filter scenes from Objaverse to select 43 high quality moderately to large sized scenes (A.1). To ensure a unified scale between scenes, we annotate scenes with scales relative to each other, aligning them under a uniform scale (A.2). Finally, we preprocess the raw Objaverse mesh files for training by sampling point clouds, adjusting ground geometry, and converting them to occupancies (A.3). Furthermore, we describe the sampling maps used to sample quad chunks from scenes (A.4) as well as some dataset statistics (A.5). Please see our dataset page `https://3dlg-hcvc.github.io/NuiScene43-Dataset` for visualizations of all 43 scenes in NuiScene43.

A.1. Filtering

We begin by filtering Objaverse [2] using multi-view embeddings from 12 object frames using DuoduoCLIP [5] . We use cosine similarity to query embeddings using text and images to retrieve scenes, then refine selection through manual labeling and neural network filtering trained on DuoduoCLIP embeddings, reducing 37k initial scenes to 2k. Since manually processing all scenes for scale labeling and ground alignment is impractical, we select 43 larger scenes for initial experiments.

A.2. Scale Labeling

To establish a uniform scale across scenes, we first normalize all scenes to $[-1, 1]$. We then randomly select one scene as the anchor, assigning it a scale of 1. Each remaining scene is compared with the anchor, and its scale is adjusted visually to align elements such as trees or buildings from multiple angles, to ensure a visually coherent relative size. Since these scenes are artist-created, their proportions may prioritize aesthetic appeal over real-world accuracy. As a result, there is no definitive correct scale. Instead, we approximate a visually consistent scaling and record the assigned scale for each scene, with the anchor scene remaining fixed at scale 1.

A.3. Geometry Processing

Point Cloud and Occupancy. All scene meshes are first converted into SDFs using the same method as Wang et al. [7], re-implemented in Taichi [4] for faster conversion. The scales obtained in A.2 are used to adjust the voxel grid resolution for SDF conversion to enforce the unified scale across scenes. We then convert to occupancy by thresholding SDF values. Finally, multiple iterations of flood filling are applied to fill in holes in the scene. Point clouds are sampled from the meshes after applying the Marching Cubes algorithm to the occupancy of each scene.

Ground Fixing. We observed that the filtered scenes had various ground geometry, ranging from flat planes to thick volumes. To deal with this, we identify the lowest ground level in each scene and enforce a uniform ground thickness of 5 voxels below this level in the occupancy grid, ensuring consistency in ground geometry across scenes.

Table 1. NuiScene43 statistics.

Category	# Scenes	# Sampleable Quad Chunks
Rural/Medieval	16	6.2 M
Low Poly City	19	4.9 M
Japanese Buildings	4	2.9 M
Other	4	1.2 M

A.4. Chunk Sampling

To sample training chunks from scenes, we first compute an alpha map from the top-down view. Next, we apply a convolution with all one kernel weights over the entire scene using a kernel size of $(100, 100)$, the size of a quad chunk along the x and z axis. The resulting convolution map is then thresholded at a value of 100×100 to determine valid sampling locations. This ensures that all sampled chunks contain occupied regions, avoiding holes or boundary areas in the scenes.

Next, we compute a depth variation map. For each pixel, we calculate the mean depth within the kernel window and subtract the pixel's depth value, taking the absolute value to obtain depth mean deviations. To avoid sampling overly flat regions, we filter out sample locations where the depth variation is below 2.5. An illustration of the sample maps is provided in Figure 2.

Finally, we apply farthest point sampling (FPS) to select quad chunks from all valid sampling locations. This helps minimize excessive overlap between chunks while ensuring maximum scene coverage for training.

A.5. NuiScene43 Statistics

We present statistics of NuiScene43 in Tab. 1, including the number of scenes in each category and the total number of quad chunks that can be sampled. The chunk count is derived by summing the sampling map discussed earlier, following depth variation thresholding. Note that these chunks may overlap, as we consider all valid x and z coordinates.

B. Implementation Details

B.1. 4-scene Training

We add an additional 3 scenes along with the original single scene for training the 4-scene model in the main paper. The 3 additional scenes are shown in Figure 3.

B.2. Raster Scan Order Generation

We show the algorithm of our raster scan order generation during inference utilizing our diffusion model trained with 4 different configurations in Algorithm 1. The algorithm outlines how the 4 different masking configurations are used to

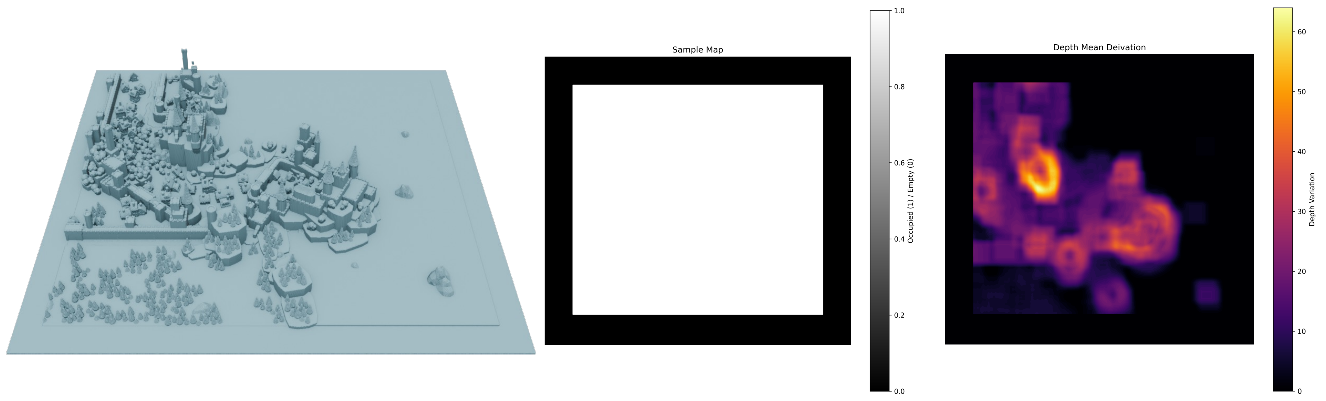


Figure 2. Sample and depth mean deviation maps are calculated for sampling chunks from scenes.

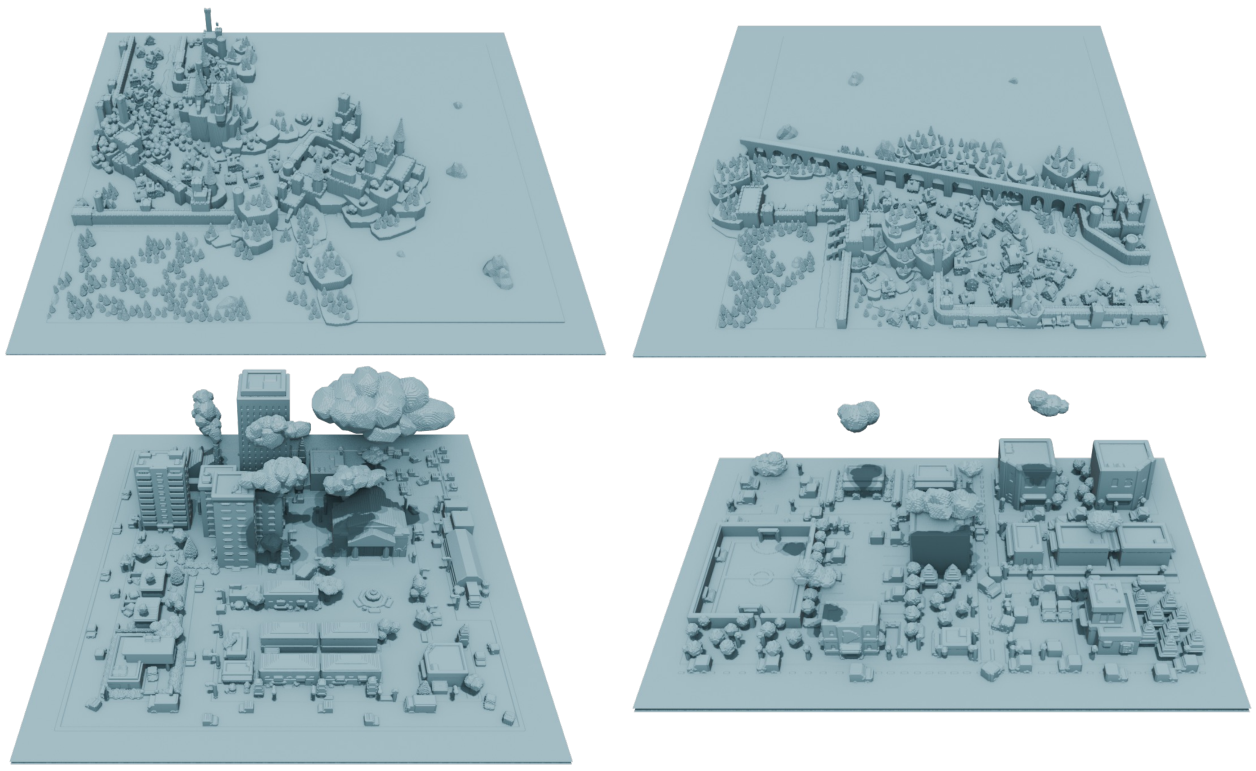


Figure 3. Three additional scenes used to train our 4-scene model. The top two sub-scenes are split from a large Objaverse scene for occupancy calculation. All scenes have fixed ground geometries, and their meshes are extracted via marching cubes on the occupancy grid.

condition on existing chunks for continuous and unbounded generation. In line 19 of the algorithm we denoise for 50 steps using the DDPM scheduler. After we obtain a large grid of embeddings, the decoder is used to decode occupancies followed by marching cubes to get the final mesh.

It can be observed that generation of the next row can begin once a single quad chunk has been generated in the current row. This allows for parallelization along the scene's anti-diagonal, greatly accelerating sampling for large scenes. However, memory usage becomes inconsistent and varies with scene size. For consistency and simplicity, we use the sequential raster scan order (Algorithm 1) in the paper. Please refer to our implementation for more details.

B.3. Details on Scene Texturing

SceneTex [1] offers three camera modes[1] for rendering images and texture optimization, i.e., *Spherical Cameras*, the

[1]https://github.com/daveredrum/SceneTex

Algorithm 1 Raster Scan Order Scene Generation

```
Require: Number of rows $I$, number of columns $J$, trained diffusion model $\epsilon_\theta$
 1: Initialize scene latent grid $\boldsymbol{Z} \in \mathbb{R}^{I \times J \times V \times c}$
 2: for $i = 0$ to $I - 2$ do                                          ▷ Iterate row-wise
 3:     for $j = 0$ to $J - 2$ do                                      ▷ Iterate column-wise
 4:         if $i = 0$ and $j = 0$ then                                ▷ First chunk (Full)
 5:             $M \leftarrow \{0, 0, 0, 0\}$
 6:             $Z_{\text{cond}} \leftarrow \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}\}$
 7:         else if $i = 0$ then                                       ▷ First row (Left-Right)
 8:             $M \leftarrow \{1, 0, 1, 0\}$
 9:             $Z_{\text{cond}} \leftarrow \{\boldsymbol{Z}_{i,j}, \mathbf{0}, \boldsymbol{Z}_{i+1,j}, \mathbf{0}\}$
10:         else if $j = 0$ then                                       ▷ First column (Top-Down)
11:             $M \leftarrow \{1, 1, 0, 0\}$
12:             $Z_{\text{cond}} \leftarrow \{\boldsymbol{Z}_{i,j}, \boldsymbol{Z}_{i,j+1}, \mathbf{0}, \mathbf{0}\}$
13:         else                                                       ▷ All other cases (Diagonal)
14:             $M \leftarrow \{1, 1, 1, 0\}$
15:             $Z_{\text{cond}} \leftarrow \{\boldsymbol{Z}_{i,j}, \boldsymbol{Z}_{i,j+1}, \boldsymbol{Z}_{i+1,j}, \mathbf{0}\}$
16:         end if
17:         $\boldsymbol{X}_T \sim \mathcal{N}(0, \mathbf{I})$                ▷ Sample Gaussian noise
18:         $\boldsymbol{C} \leftarrow M \oplus Z_{\text{cond}} \oplus \text{PE}$   ▷ Conditioning input
19:         $\{\boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3\} \leftarrow \text{Denoise}(\boldsymbol{X}_T \oplus \boldsymbol{C}, \epsilon_\theta)$   ▷ Denoise 2×2 quad chunk
20:         $\boldsymbol{Z}_{i,j}, \boldsymbol{Z}_{i,j+1}, \boldsymbol{Z}_{i+1,j}, \boldsymbol{Z}_{i+1,j+1} \leftarrow \boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3$   ▷ Write to scene latent grid
21:     end for
22: end for
23: return $\boldsymbol{Z}$
```

Blender Cameras and *BlenderProc Cameras*. For small indoor scenes like those from 3DFront [3], spherical cameras are sufficient to capture the fine-grained details of the scene and produce high-quality textures. However, for large outdoor scenes, a predefined spherical camera trajectory often fails to cover the entire scene comprehensively and misses important scene details. We therefore choose the *Blender Cameras* mode to manually keyframe the camera trajectory for each scene. This enables a finer control over capturing the scene details, leading to an improved texture.

More specifically, we first normalize each scene into a spatial range of $[-1, 1]^3$ centered at the world origin. Then, we define a snake-scan-like camera trajectory to capture the entire scene at a specified frame rate, as shown in Figure 4. Each frame will be rendered by SceneTex and all rendered images are sampled during training to optimize the scene-level texture. Note that to enable faster optimization, we use Blender's *Decimate Geometry* and *Merge By Distance* with a distance of 0.001m to compress the scene until its size is smaller than 20MB. This might degrade the geometry for large scenes, but it's only for the texturing purpose. We encourage readers to focus on the untextured geometry details of the generated scenes. Figure 5 shows more examples on raw geometries and the textured counterparts.

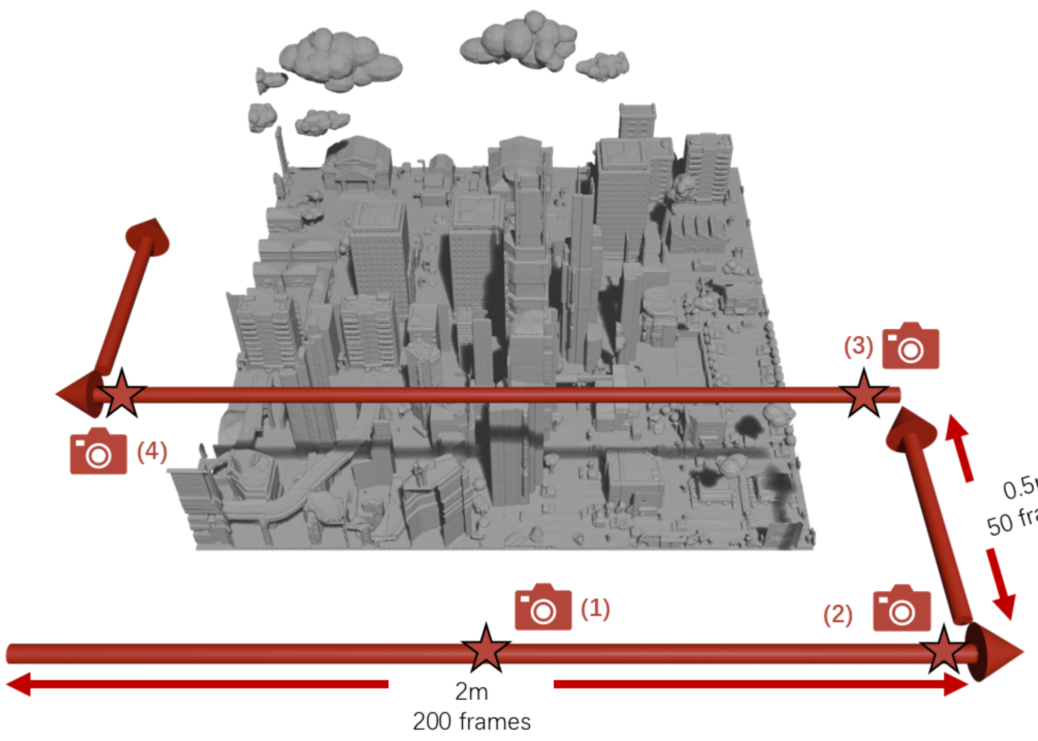
Table 2. Additional quantitative comparison of reconstruction across different VAE backbones for the single scene experiment. Here $\hat{h}$ indicates that the predicted height was used for the occupancy prediction and $\tilde{h}$ the ground truth height.

Method	Output Res/S	IOU↑	CD $(\hat{h})\downarrow$	F-Score $(\hat{h})\uparrow$	CD $(\tilde{h})\downarrow$	F-Score $(\tilde{h})\uparrow$
triplane	$3 \times 64^2/6$	0.940	0.064	0.831	0.064	0.831
	$3 \times 128^2/6$	0.982	0.185	**0.879**	0.058	0.858
vecset	-	**0.989**	**0.055**	0.864	**0.055**	**0.863**

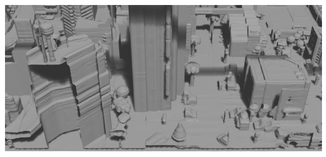
C. Additional Results

C.1. Single Scene Results

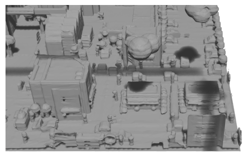
Triplane 3×128^2. Tab. 2 presents results for the Triplane baseline with an output resolution of 3×128^2. We see that the IOU outperforms the lower-resolution triplane baseline and beats the vecset model on F-Score($\hat{h}$). However, we notice a large increase in CD($\hat{h}$). Upon inspecting chunks with larger CD values, when predicted heights are larger than the ground truth, the model produced floating artifacts. This may be attributed to the higher resolution, which may be more sensitive to spatial aliasing or quantization artifacts during feature querying. Adjusting the scale factor to $S = 8.5$ may be more appropriate for this resolution, but due to the high training cost, we did not retrain the model. These results highlight the sensitivity of spatially structured latents to scene bounds and hyperparameters like S.



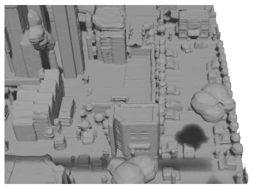
(a) An illustration of our choice for the camera trajectory, indicated by red arrows.



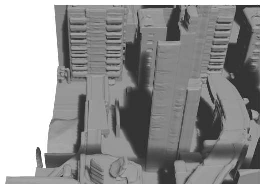
(b) Image rendered at position (1).



(c) Image rendered at position (2).



(d) Image rendered at position (3).



(e) Image rendered at position (4).

Figure 4. An overview of our choice of the camera trajectory in Blender and the four images respectively rendered at position (1), (2), (3) and (4). We adopt a snake-scan trajectory pattern allowing for a more comprehensive coverage of the entire scene. The long side of the trajectory spans 2 meters at a fixed number of 200 frames, and the shorter side spans 0.5 meters for 50 frames. Depending on the shape of the scene, the total number of frames ranges from 1.2k to 1.8k.

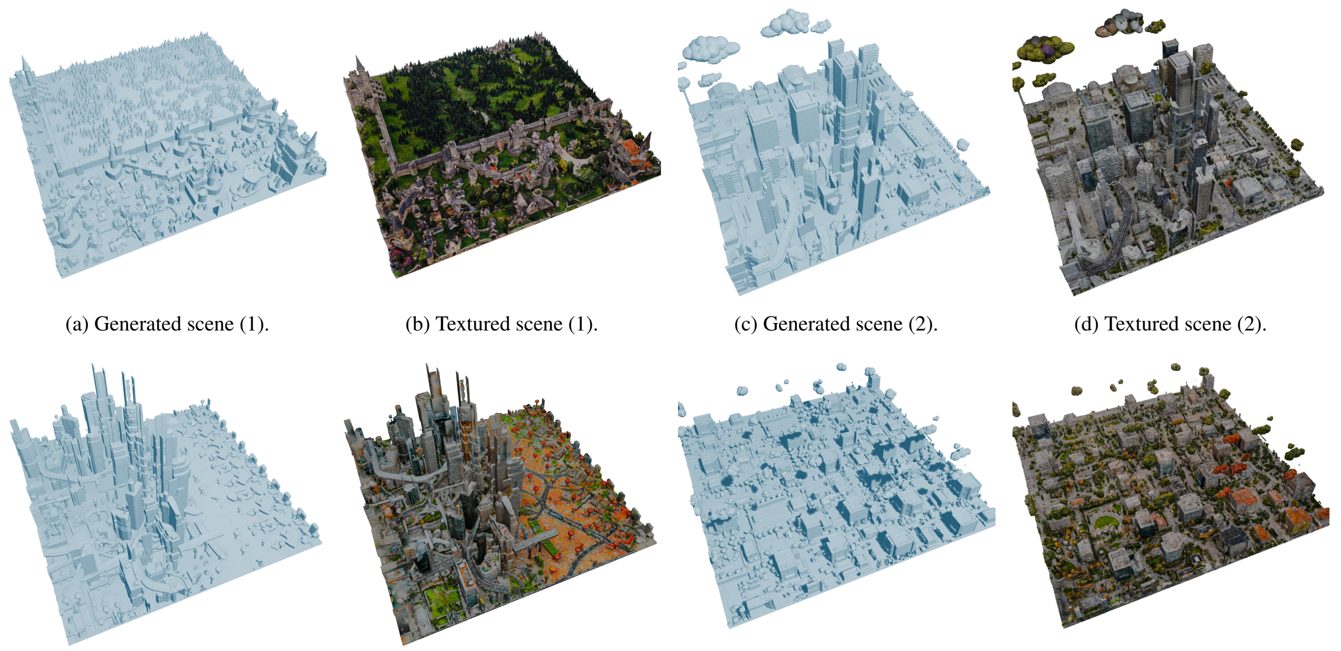
Table 3. Comparison of VAE training resources for vector set vs triplane backbones with larger batch sizes. Training for all experiments was run on 4 L40S GPUs, total batch size and memory across 4 gpus are reported. The # Latents is the size of the latent for the VAE backbone and Output Res indicates the triplane size after deconvolution.

Method	BS	# Latents	Output Res	Time (hr)	Mem. (GB)
triplane	192	3×4^2	3×32^2	18	166.0
	120	3×4^2	3×64^2	29.4	164.4
vecset	144	16	-	21.6	170.5

Table 4. Quantitative comparison of reconstruction across different VAE backbones using larger batch sizes in Tab. 3. Here $\hat{h}$ indicates that the predicted height was used for the occupancy prediction and $\tilde{h}$ the ground truth height.

Method	Output Res/S	IOU↑	CD ($\hat{h}$) ↓	F-Score ($\hat{h}$) ↑	CD ($\tilde{h}$) ↓	F-Score ($\tilde{h}$) ↑
triplane	$3 \times 32^2/6$	0.727	0.171	0.508	0.170	0.503
	$3 \times 64^2/6$	0.933	0.099	0.852	0.064	0.830
vecset	-	**0.962**	**0.072**	**0.890**	**0.057**	**0.858**

Larger Batch Size. We initially experimented with larger batch sizes across all configurations, using 4 GPUs, as shown in Tab. 3 to accelerate training. For the vecset model, this reduced training time to 21.6 hours from 36.1 hours on 2 GPUs (main paper configuration). However, it led to overfitting, as seen in Tab. 4 which also affected height prediction accuracy as demonstrated in Figure 6 where the model under-predicted the height, leaving parts of the chunk unreconstructed. In contrast, triplane backbones were less affected by overfitting. We hypothesize this is due to their spatial locality where query points retrieve features from



(a) Generated scene (1). (b) Textured scene (1). (c) Generated scene (2). (d) Textured scene (2).

(e) Generated scene (3). (f) Textured scene (3). (g) Generated scene (4). (h) Textured scene (4).

Figure 5. Four examples (1) to (4) of generated scenes and the textured counterparts. (1) is textured using prompt "A beautiful town with buildings, castles and trees". (2)(3) are textured using prompt "A large city with buildings and surroundings". (4) is textured using prompt "A modern city with buildings and trees, clouds in the sky".

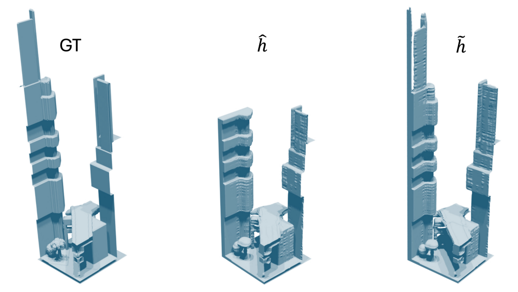


Figure 6. We compare the reconstruction of the vector set model with larger batch size (144) when using the predicted height $\hat{h}$ of the model and using the ground truth height $\tilde{h}$.

fixed triplane positions, whereas the vecset backbone allows queries to aggregate from all feature vectors, offering greater capacity but also more prune to overfitting. Nevertheless, this shows the limit of the one scene training.

Evaluation Metrics. Following 3DShape2Vecset [8], we report IOU, Chamfer Distance (CD), and F-Score for VAE reconstruction. However, we find that CD and F-Score can be less reliable. When extracting surfaces via marching cubes, a level set threshold must be selected, which can introduce gaps relative to the ground truth due to the discrete grid resolution. This issue can disproportionately affect chunks of varying height, taller chunks may have fewer points per area given the same number of points are sampled for each chunk, leading to potentially higher CD and lower F-Score. As a result, these metrics are not fully comparable across settings like single-scene versus 4-scene training, due to differences in chunk distributions. In our case, larger differences in reported values are more indicative, while smaller variations may reflect noise due to the level set gap. Overall, IOU is more reliable and comparable across different scene configurations.

Level of Detail. In Figure 7, we visualize occupancy predictions at different resolutions during inference with marching cubes to generate different level of detail (LoD) for the triplane and vector set models. 50 is the original chunk size used for training along the x and z axis, with the y axis scaled according to the height prediction. We find that halving the chunk size offers comparable details while reducing memory usage for scenes. However, decreasing further to 13 leads to noticeable loss of details in the trees and buildings.

Additional Visualizations We show additional results for our vecset diffusion model in Figure 8 and results for the RePaint baseline in Figure 9.

C.2. 4-scene

For the 4-scene experiments in this section we follow the same GPU configurations in the main paper, and increase the number of sampled chunks across 4 scenes to 300K chunks.

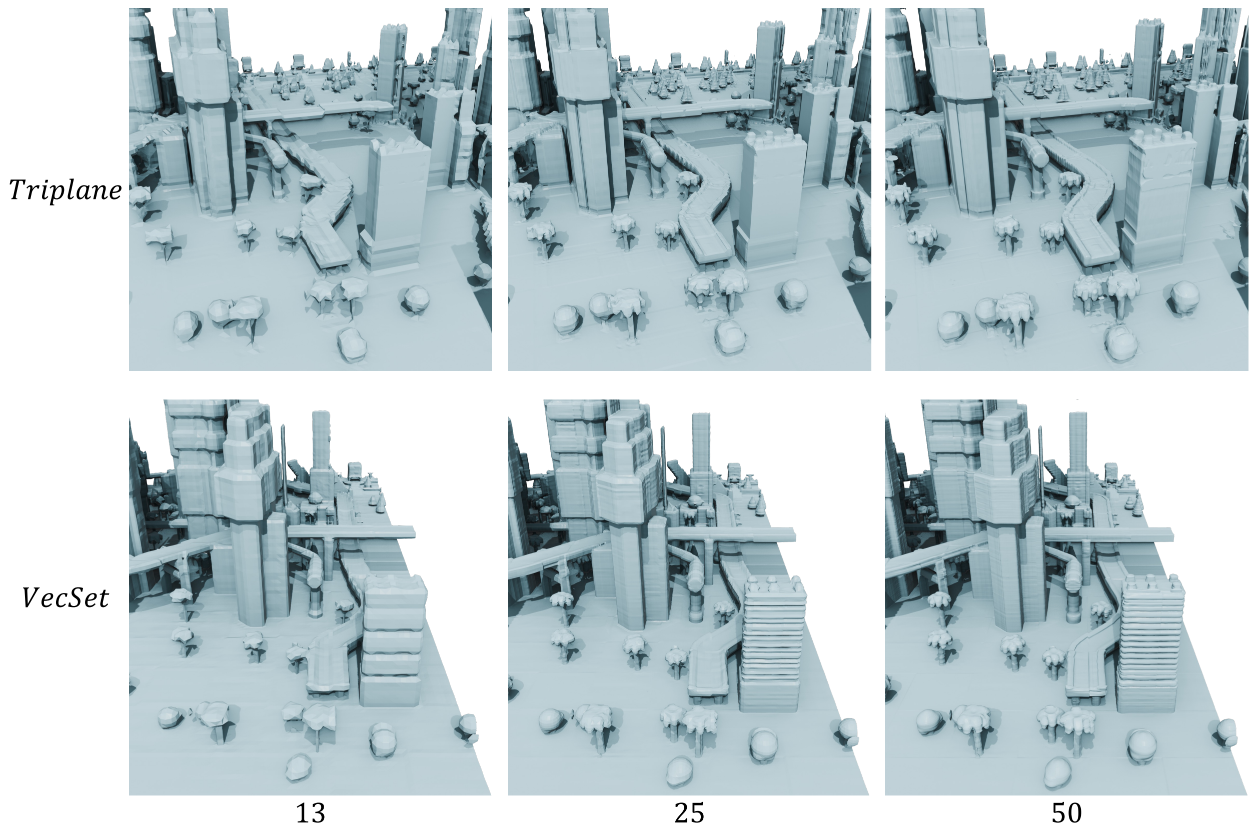


Figure 7. Here we show the results of using different occupancy grid resolutions for prediction during inference for marching cubes. The numbers on the bottom indicate the chunk size along the x and z axis. 50 is the original chunk size used for training.

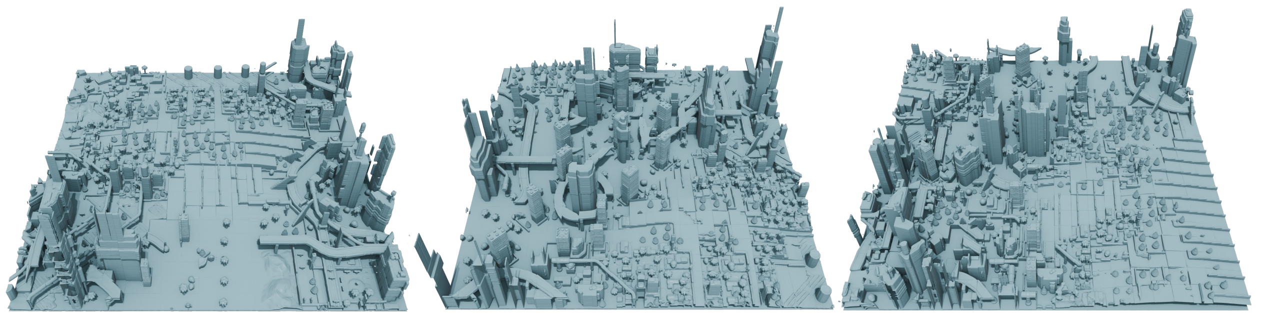


Figure 8. Additional results for our vector set diffusion model trained on a single scene. The scenes shown here are of size 21×21.

Table 5. Quantitative comparison of reconstruction across different VAE backbones on the 4 scene training. Here $\hat{h}$ indicates that the predicted height was used for the occupancy prediction and $\tilde{h}$ the ground truth height.

Method	Output Res/S	IOU↑	CD ($\hat{h}$) ↓	F-Score ($\hat{h}$) ↑	CD ($\tilde{h}$) ↓	F-Score ($\tilde{h}$) ↑
triplane	$3 \times 64^2/6$	0.933	0.061	0.833	0.061	0.832
vecset	-	**0.989**	**0.053**	**0.869**	**0.053**	**0.868**

Table 6. Comparison of triplane and vecset diffusion models for generated quad-chunks on the 4 scene training. KPD scores are multiplied 10^3.

Method	FPD↓	KPD↓
triplane	0.868	2.142
vecset	0.581	1.104

The training and validation follow the same 95%-5% split. For diffusion evaluation we sample 30K quad chunks from the scenes and well as the diffusion models for evaluation.

Quantitative Results. We show the VAE reconstruction and

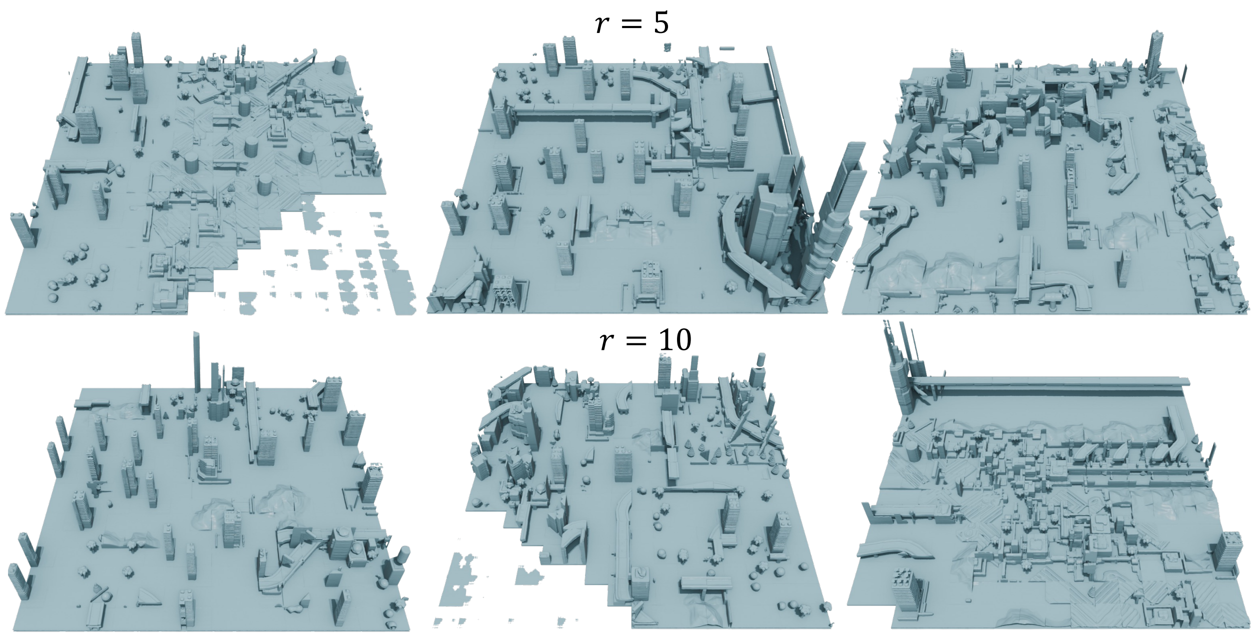


Figure 9. Additional RePaint results using our vector set diffusion model trained on a single scene. The scenes shown here are of size 16×16. We show results for resampling steps $r = 5$ and $r = 10$. Compared to our outpainting model, RePaint struggles with inter-chunk coherence and sometimes collapses, producing broken chunks in larger scenes.

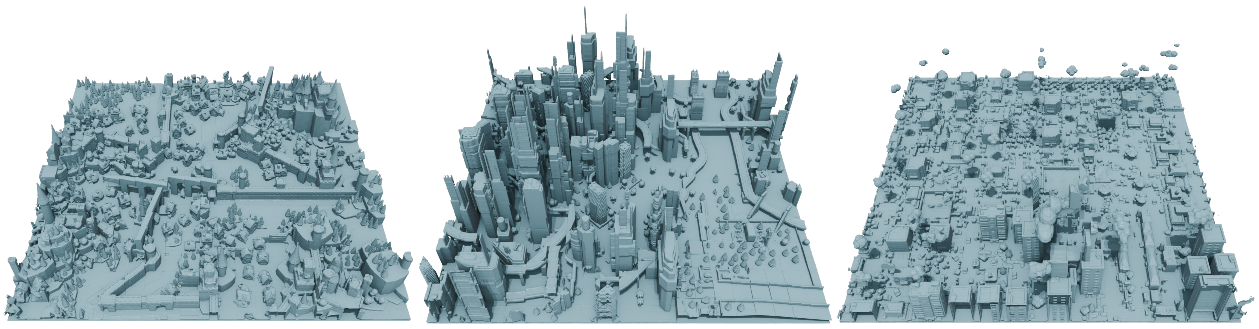


Figure 10. Additional results for our vector set diffusion model trained on 4 scenes. The scenes shown here are of size 21×21.

diffusion quality in Tab. 5 and Tab. 6, respectively. We can see that for IOU the vector set model maintains its performance, while the triplane slightly drops. With the vector outperforming the triplane model across the board.

Additional Visualizations. We show additional results from our vecset diffusion model trained on 4 scenes in Figure 10.

C.3. 13-scene

To further demonstrate our model's potential for scaling up, we sample 280K chunks from 13 scenes in NuiScene43 (sample configurations available here) and train using the same settings as before.

Improving Reconstruction. Following the previous vector set settings, the VAE model trained on 13 scenes see a drop in IOU to 0.968, due to a larger variation of scenes compared to the single or 4-scene scenarios. One naive way we can increase the VAE's performance is to increase the number of feature vectors directly during training from $V = 16$. However, this would increase the memory usage and slow diffusion training as shown in the triplane model. Instead, inspired by the deconvolution layers for triplanes. We introduce a pixel shuffle [6]-like layer for upsampling the number of vector sets followed by additional self attention layers. This increases the capacity of the model without increasing the latent size. Specifically, we add an additional projection layer that increases the number of channels and reshape from the channels dimension to the vector tokens. We upsample to 512 vector sets and add 3 self-attention layers before the cross attention layer for querying coordinates. This improves the IOU to 0.983.

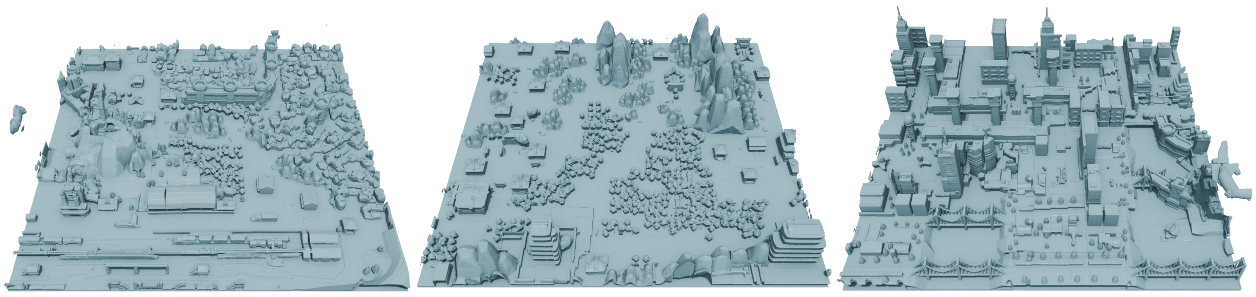


Figure 11. Additional results for our vector set diffusion model trained on 13 scenes. The scenes shown here are of size 21×21.

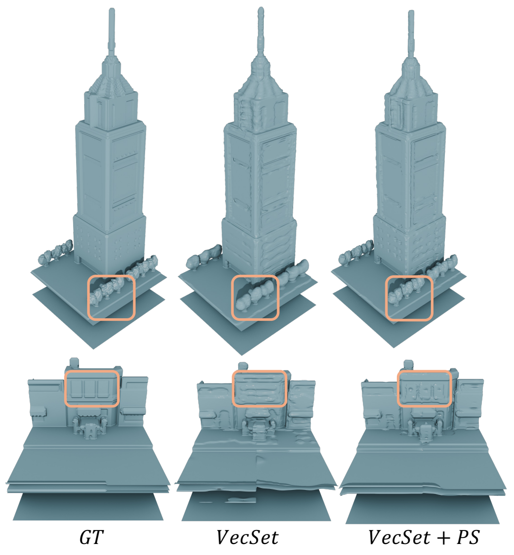


Figure 12. We compare chunk reconstructions between the original vector set model and the upsampled version (VecSet + PS). The upsampling model recovers finer details that are blurry in the original (see orange boxes).

In Figure 12, we show a qualitative comparison between models. The original vector set model already reconstructs scenes quite well, despite the increase in the number of training scenes. The added improvements further enhance the reconstruction of finer details.

Additional Visualizations. We show square scenes generated by the diffusion model trained with the improved model with pixel shuffle in Figure 11. And larger scenes with aerial and street view zoom-ins in Figure 1. It can be seen that this model can generate a larger variety of different scenes while maintaining good fidelity.

References

[1] Dave Zhenyu Chen, Haoxuan Li, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. SceneTex: High-quality texture synthesis for indoor scenes via diffusion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21081–21091, 2024. 3

[2] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 2

[3] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3D-front: 3D furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021. 4

[4] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):201, 2019. 2

[5] Han-Hung Lee, Yiming Zhang, and Angel X Chang. Duoduo CLIP: Efficient 3D understanding with multi-view images. *arXiv preprint arXiv:2406.11579*, 2024. 2

[6] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 8

[7] Peng-Shuai Wang, Yang Liu, and Xin Tong. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Transactions on Graphics (TOG)*, 41(4):1–15, 2022. 2

[8] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023. 6