InfiniCity: Infinite-Scale City Synthesis

Chieh Hubert Lin[1,2] Hsin-Ying Lee[2] Willi Menapace[2,3] Menglei Chai[2]
Aliaksandr Siarohin[2] Ming-Hsuan Yang[1,4,5] Sergey Tulyakov[2]

[1]UC Merced, [2]Snap Inc., [3]Trento University, [4]Yonsei University, [5]Google Research

https://hubert0527.github.io/infinicity/

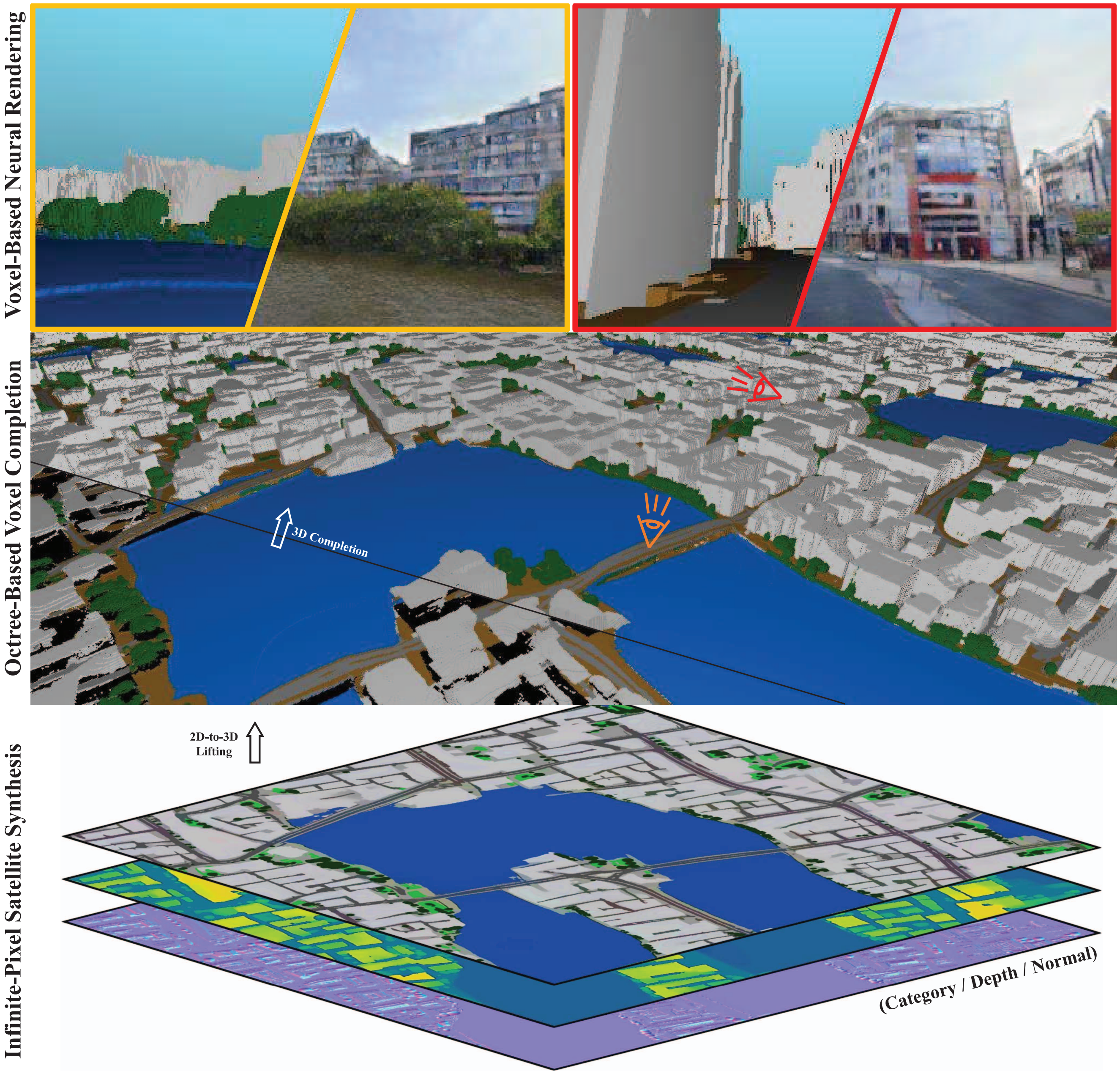
Voxel-Based Neural Rendering
Octree-Based Voxel Completion
3D Completion
Infinite-Pixel Satellite Synthesis
2D-to-3D Lifting
(Category / Depth / Normal)

Figure 1: **We propose InfiniCity, a three-stage synthesis framework toward infinite-scale city scene synthesis.** Starting from the bottom to the top, we synthesize multi-modality infinite-pixel satellite images, perform octree-based voxel completion to create a watertight voxel world, then finally texturize with voxel neural rendering. In the middle figure, we mark the camera locations (in red and orange) used to render the views in the top figures.

Abstract

Toward infinite-scale 3D city synthesis, we propose a novel framework, InfiniCity, which constructs and renders an unconstrainedly large and 3D-grounded environment from random noises. InfiniCity decomposes the seemingly impractical task into three feasible modules, taking advantage of both 2D and 3D data. First, an infinite-pixel image synthesis module generates arbitrary-scale 2D maps from the bird's-eye view. Next, an octree-based voxel completion module lifts the generated 2D map to 3D octrees. Finally, a voxel-based neural rendering module texturizes the voxels and renders 2D images. InfiniCity can thus synthesize arbitrary-scale and traversable 3D city environments. We quantitatively and qualitatively demonstrate the efficacy of the proposed framework.

1. Introduction

With the rapid evolution in the generative modeling research, modern generators can synthesize high-quality images with high-fidelity [13, 21, 34], 3D consistent content with neural rendering [6, 16, 17], and temporal-consistent videos [4, 19, 27]. However, most of these works focus on perfecting the synthesis quality with limited camera movement or within a bounded space, therefore less suitable for modeling an unconstrainedly large scene. Recent attempts to achieve infinite visual synthesis with neural implicit model [24] or connecting anchors [40] are only feasible for the side view of landscapes or city scenes and create unrealistic global structures at extreme-large field-of-views, while another effort toward navigable 3D indoor scene synthesis [12] is designed for bounded environments with a constrained camera distribution and the synthesis is of limited resolution. These limitations pique our interest in synthesizing infinite-sized, realistic, and navigable 3D environments.

Among all 3D environments, we take city scenes as a case study as they are ubiquitous in contemporary gaming, virtual reality, and augmented reality as an everyday sight. Ideally, it is desirable and straightforward to synthesize the whole 3D environment all at once. However, it is currently impractical with existing techniques and hardware constraints. We propose breaking down the synthesis of a 3D environment into stages of global structure planning and local perfection. First, intuitively, the satellite view of a city illustrates the outlines of the city, and provides abundant clues about the major structural information. We can leverage this information to plan the global structure. As the remaining missing information is local, we can therefore finish the structure with local 3D completion, then finalize the textures with view-dependent neural rendering.

Based on these observations, we develop InfiniCity, an effective pipeline toward infinite-scale 3D environment synthesis embracing the benefits from both 2D and 3D data. As shown in Figure 1, InfiniCity consists of three major modules. First, we use patch-based implicit image synthesis [24] to learn how to generate arbitrary-scale 2D maps from bird's-eye views of 3D data. Second, we employ an memory- and computation-efficient octree representation to lift the 2D maps to a 3D representation using a 3D completion module. Finally, we perform neural rendering [17] on the generated octrees using real 2D images as well as pseudo-realistic images generated via state-of-the-art semantic synthesis method [31].

In this work, we present the first attempt at synthesizing an unbounded environment on an infinite scale. InfiniCity is a sophisticated framework breaking down the originally improbable task into sub-modules. We conduct quantitative and qualitative experiments on the HoliCity dataset [60] to corroborate the necessity and validate the effectiveness of the multi-stage pipeline using both 2D and 3D data. We also perform ablation studies on various pivotal design choices and pivotal operations, including patch contrastive learning and bilateral filtering.

2. Related Work

Infinite Visual Synthesis Several recent attempts explore modeling infinitely large environments using only finite images with limited field-of-view. One direction is to generate arbitrary-scale static images with a divide-and-conquer strategy where small patches are synthesized and then merged [23]. The inference process can either be autoregressive [14, 51] or non-autoregressive [24, 40]. Another branch of work focuses on the long-range generation of novel views along with a camera trajectory [22, 25]. However, the videos rendered by such an approach lack 3D consistency, since the appearances are modeled in the 2D image space without utilizing the 3D representations. In this work, we explore the infinite synthesis in the 3D city scene, aiming to generate a 3D-grounded traversable environment of infinite scale.

It is worth noting a few concurrent works also explore unbounded 3D scene generation under different camera assumptions. SGAM [36] proposes to render the surface of the 3D scene from pure satellite view. PersistentNature [5] and SceneDreamer [8] explores camera fly-through video similar to InfiniteNature [25]. However, none of these works focus on the within-the-scene navigable scene generation similar to our setting.

3D Generation from 3D Data To learn a 3D generative model, it is intuitive to train it on 3D data. Recent works have explored various 3D representations, including point clouds [2, 26], voxels [42, 52], signed distance functions [9, 10, 11, 30], etc. In this work, we also leverage explicit 3D

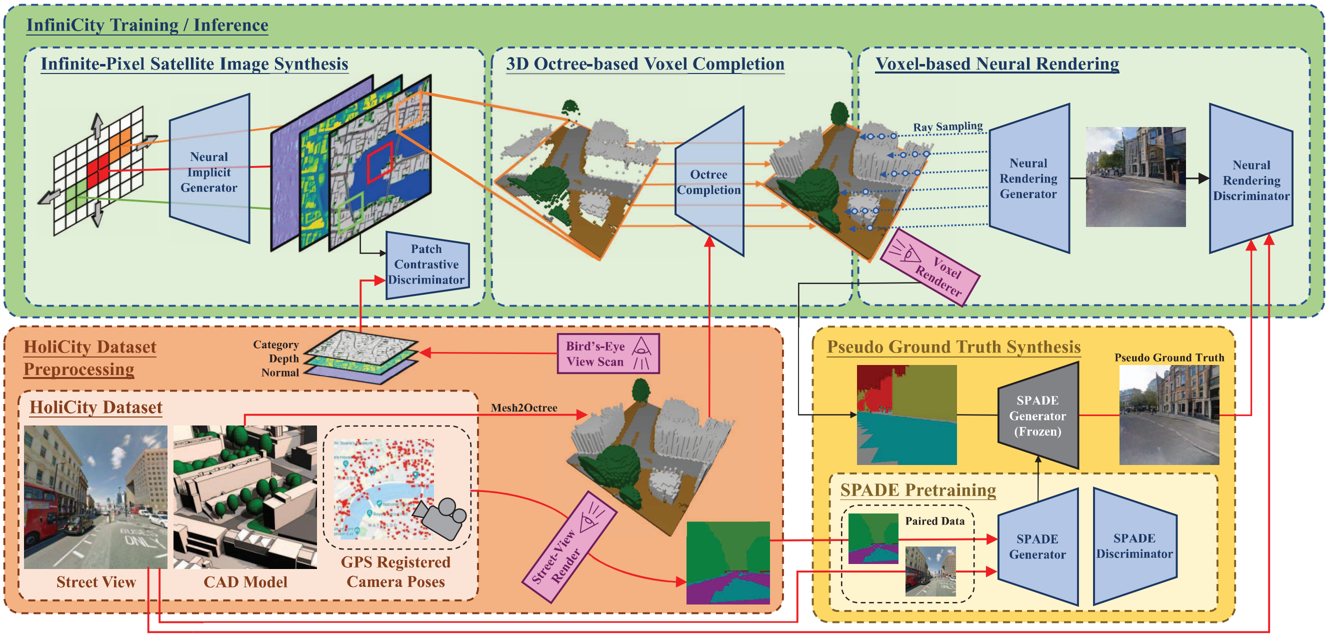
InfiniCity Training / Inference
Infinite-Pixel Satellite Image Synthesis
Neural Implicit Generator
Patch Contrastive Discriminator
3D Octree-based Voxel Completion
Octree Completion
Voxel-based Neural Rendering
Ray Sampling
Neural Rendering Generator
Neural Rendering Discriminator
Voxel Renderer
HoliCity Dataset Preprocessing
Category
Depth
Normal
Bird's-Eye View Scan
HoliCity Dataset
Street View
CAD Model
GPS Registered Camera Poses
Mesh2Octree
Street-View Render
Pseudo Ground Truth Synthesis
Pseudo Ground Truth
SPADE Generator (Frozen)
SPADE Pretraining
Paired Data
SPADE Generator
SPADE Discriminator

Figure 2: **Overview.** InfiniCity consists of three major modules. The ***Infinite-pixel satellite image synthesis*** stage is trained on image tuples (category, depth, and normal maps) derived from a bird's-eye view scan of the 3D environment, and is able to synthesize arbitrary-scale satellite maps during inference. The ***3D octree-based voxel completion*** stage is trained on pairs of surface-scanned and completed octrees. During inference, it takes the surface voxels lifted from the satellite images as inputs and produces the watertight voxel world. Finally, the ***voxel-based neural rendering*** stage performs ray-sampling to retrieve features from the voxel world, then renders the final image with a neural renderer. The neural renderer is trained with both real images and pseudo-ground-truths synthesized by a pretrained SPADE generator. With these modules, InfiniCity can synthesize an arbitrary-scale and traversable 3D city environment from noises.

supervision to learn the geometry of the 3D environment. Specifically, we adopt octree [33, 46, 49, 48], a sparse-voxel representation, as our 3D representation.

3D Generation from 2D Data Inspired by Neural Radiance Fields (NeRF) [29], NeRF-based generators [6, 7, 16, 35, 37, 39, 41, 53] combined with GAN-based framework [15, 21] have become a dominating direction to learn 3D structure from 2D image collections. These methods are restricted to modeling single objects within a limited range of viewing angles from a camera. Relaxing the constraints, GSN [12] models traversable indoor scenes with many local radiance fields, yet it requires trajectories of calibrated images for training and is not directly applicable to city scenes due to the unbounded nature and complexity of outdoor scenes. The other branch of work targets performing texturization on a given 3D representation. Given a 3D object [38], a 3D scene [20], or a voxel world [17], colorization and texturization are learned with differentiable rendering followed by a GAN-based framework.

3. InfiniCity

We aim to generate infinite-scale 3D city scenes using both 2D and 3D data. In Figure 2, the InfiniCity synthesis pipeline consists of three main components. The infinite 2D-map synthesis first generates an arbitrarily large satellite map from random noises, the octree-based voxel completion model converts the map into a watertight voxel environment, then neural rendering texturizes the voxel world. We further discuss each of the components.

3.1. Data Preprocessing

The dataset consists of images with GPS-registered camera poses I, p and CAD model C representing the scenes. We further process the data for each of the three modules. For the octree-based voxel completion, we first convert and partition the CAD model to a set of octrees $\{O_i\}$, each representing a sub-region of the city. These octrees are not only the supervision for the 3D completion training but also further render into two different types of training data.The "bird's-eye view scan" extracts multiple modalities of the octree surface information into surface octrees $\{O_i^{\text{sur}}\}$ from the top-down direction, then these surface octrees are further converted into 2D images (i.e., categorical, depth, and normal maps), jointly denoted as I^{CDN}. The $\{O_i^{\text{sur}}\}$ pairing with $\{O_i\}$ constitutes the 3D completion model training data, while I^{CDN} serves as the training data for the infinite-pixel satellite image synthesis. On the other hand, the "street-view render" utilizes the GPS-registered camera location along with annotated camera orientation p_j to render segmentation images I_j^{seg} corresponding to the street-view images I_j. Such procedure constructs data pairs for training SPADE [31], which later synthesizes the pseudo-ground-truth during the neural rendering training.

3.2. Infinite 2D Map Synthesis

Infinite-scale generation directly on 3D data is currently far-fetched, yet recently explored on 2D images. Therefore, instead of directly generating the 3D environment, we propose to start by synthesizing the corresponding 2D map. We leverage the infinite-pixel image synthesis ability of InfinityGAN [24], which synthesizes arbitrarily large maps with the neural implicit representation. Due to the limitations of data, we generate categorical labels instead of real RGB satellite images. However, as GANs encounter problems in propagating gradients while modeling discrete data [56, 58], we instead assign colors to each of the classes and train InfinityGAN on the categorical satellite map rendered with the assigned colors. The colors are later converted back to a discrete category map with the nearest color. Meanwhile, to convert the predicted satellite images to a voxel surface for the next-stage 3D shape completion, we jointly model the height map information. To further regularize the structural plausibility, we further model the surface normal vector, which is the aggregated average surface normal over the unit region covered by a pixel in the satellite view.

In InfiniCity, a critical problem is that the satellite image has a larger field-of-view, compared to the original InfinityGAN setting. Accordingly, InfiniCity requires extra focus on the structural plausibility in the local region. Directly applying adversarial learning on a large and dense matrix makes the discriminator focus on global consistency and overall visual quality, instead of the local details. Therefore, we apply the contrastive patch discriminator [32] to increase the importance of the fine-grained details.

We synthesize tuples of images of arbitrary scale in this stage: $\hat{I}^{\text{CDN}} = G_\infty(z)$, where all the inputs and outputs of $G_\infty(\cdot)$ can be of arbitrary spatial dimensions.

3.3. Voxel World Completion

The voxel world completion stage aims to create a watertight 3D representation from the synthesized maps $\hat{I}^{\text{CDN}}$, as the following neural rendering involves ray-casting and requires reasonable ray-box intersection in the 3D space. A critical issue of voxel representation is its immense memory consumption, due to allocating unnecessary memory to the unused empty spaces. We utilize an octree-based voxel representation [48, 47] to avoid the memory issue, and implement the 3D completion model with O-CNN [47] framework for efficient neural operations directly on octrees. To retain the surface information, we build skip connections using OUNet [48], trained with voxels of spatial size 64^3. The model is trained with the paired data $\{O_i, O_i^{\text{sur}}\}$. At inference time, we partition the maps $\hat{I}^{\text{CDN}}$ generated in the previous stage into patches of 64^2 pixels, convert them into surface voxels in octree representation $\hat{O}_i^{\text{sur}}$, and obtain 3D-completed voxels via $\hat{O}_i = G_{\text{vox}}(\hat{O}_i^{\text{sur}})$ for each patch. As a spatially contiguous city surface is already illustrated by the satellite view, we empirically observe that the separately processed octree blocks remain contiguous after the 3D completion and subsequent spatial concatenation.

Despite the outputs being already visually plausible while using the raw output from the satellite-image synthesis step, we can still observe some isolated pixels caused by artifacts generated in the depth channel. These artifacts create floating voxels after converting the satellite image to the surface voxels, and further lead to undesirable structures after applying the 3D completion model. We employ bilateral filters [45] to suppress these noises. The filter is applied multiple times with different space and color thresholds. We first apply large kernels with small color thresholds to create sharper edges for the buildings, then small kernels with large color thresholds to remove isolated pixels.

3.4. Texturization via Neural Rendering

Finally, with a 3D-completed and watertight voxel environment, we can cast rays and utilize neural rendering to produce the final appearance of the realistic 3D environment. Here, we render the texturized images using GANcraft [17], a state-of-the-art neural rendering framework that has shown success in large-scale outdoor scene synthesis. Following its training paradigm, we first train a SPADE [31] model with paired segmentation and realistic street-view images $\{I_j^{\text{seg}}, I_j\}$, and use it to create pseudo-ground-truths $\{I_k^{\text{pseudo}}\}$ given segmentation maps sampled by the street-view renderer using random camera poses p_k. The watertight octrees patches $\{\hat{O}_i\}$ are concatenated and converted into the GANcraft parametrized voxel representation $\hat{V}$, where each of the voxels is parameterized by the parameters attached to its eight corners. Then, for each of the *valid* camera views, we cast view rays and extract the per-pixel trilinear-interpolated features base on the ray-box intersection coordinates in the 3D space. The neural-rendering model G_{neural} is trained with randomly paired real images I_j and pseudo images I_k^{pseudo} produced with camera poses $\{p_k\}$, and renders the synthesized images $\{\hat{I}_k\}$ based on the features retrieved with $\{p_k\}$. Following GANcraft [17], to handle the sky rendering, G_{neural} has two separate MLPs, one for rendering pixels that the rays intersect with voxels, and one for the rays escaped from the scene. These two

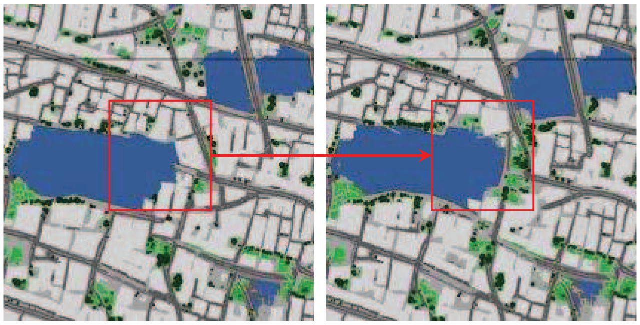

Figure 3: **Interactive resampling.** Our GUI allows users to select a region of interest and resample the local variables with efficient on-demand inference operated only on the neighbor regions.

features are later alpha-blended, and render the final $\{\hat{I}_k\}$.

However, a critical issue arises when it comes to sampling the *valid* $\{p_k\}$. To match the training distribution of the SPADE model, which provides the important pseudo-ground-truth for each camera view, we sample the camera near the ground instead of the simple fly-through camera used in GANcraft. Such a deviation leads to another issue. A city scene is occupied by lots of buildings and trees, thus it is important to detect unwanted collisions between the camera and objects. In practice, we manually select several *walkable* classes (e.g., road, terrain, bridge, and greenspace), and mark these voxels from the satellite view, forming a walkable map. As SPADE has poor performance with low entropy inputs (e.g., directly facing a wall with a uniform class), we apply three steps of erosion and connected component labeling on the walkable map, removing the small alleys and the small squares hidden between buildings. We sample the camera locations with such labeled zone, along with randomly sampled camera orientations[1]. We also observe the training becomes less stable and sometimes produces spiking gradients, thus an R1 regularization [28] is added to stabilize the training.

In summary, we can formulate the overall infinite-scale 3D scene generation process as:

Section 3.2: $\hat{I}^{\text{CDN}} = G_\infty(z)$,

Section 3.3: $\{\hat{O}_i^{\text{sur}}\} = \text{convert}(\hat{I}^{\text{CDN}}), \{\hat{O}_i\} = G_{\text{vox}}(\{\hat{O}_i^{\text{sur}}\})$,

Section 3.4: $V = \text{aggregate}(\{\hat{O}_i\}), \{\hat{I}_k\} = G_{\text{neural}}(V, \{p_k\})$.

Within the final texturized 3D environment V, we can traverse and visualize scenes given desired camera trajectories p_i via neural rendering.

3.5. Interactive Sampling GUI

Despite our pipeline synthesizing realistic satellite maps, human preferences (e.g., undesired road plans) also play an important role while assessing the quality and usability of a map. Considering our maps are infinitely large, it is unreasonable to recreate the whole map after finding disfavored local structures, where the new map still inhibits the risk of finding other problems. To resolve the issues and enable responsive interaction, we develop an interactive sampling GUI, which resamples local latent variables and randomized noises based on user control, giving the imperfect images a second chance instead of completely discarding them. As shown in Figure 3, the user can select the desired regions to resample. All the latent variables in InfinityGAN are available for resampling.

In particular, to boost interactiveness, we develop a sophisticated queuing system for efficient inference. InfinityGAN has demonstrated its ability in "spatially independent generation", where the generator can independently generate the image patches without accessing the whole set of local latent variables. Such a characteristic further enables us to queue each patch synthesis task as a job in a FIFO queue, and run inference in a batch manner by tensor-stacking multiple jobs in the queue. Furthermore, when a user selects a region of interest that intends to resample, we implement a feature calibration mechanism to collect only the subset of variables that has any contribution to the pixels within the selected region, then the subset of variables are resampled and pushed to the FIFO queue. As such, we perform only the necessary computations with the maximum GPU utilization rate, increasing the inference speed by a large margin. For instance, while interacting with a 4096×4096 map, the original synthesis method takes 16 seconds on a single RTX2080 Ti. In the case of selecting a 256×256 region, our mechanism takes only 1.7 seconds for locally resampling latent variables, which is a 10 times speed up.

4. Experiments

4.1. Dataset Processing

With the aforementioned InfiniCity algorithm, here we discuss how to extract the corresponding data modalities. HoliCity [60] is a large-scale dataset based on the 3D London CAD model (with object-level category annotations) collected by Accucities [1] and Google street-view images. The dataset contains 50,024 images registered to the CAD model with GPS location along with camera orientations. Due to the limited data accessibility, we obtain a subset of the CAD model at the L2 level of details, and the region corresponds to 14,612 registered images. We use this subset to train and evaluate our algorithm.

We first perform point sampling on the meshes surface at four times voxel resolution (i.e., sampling 4^3=64 points in each voxel grid on average). Points are sampled at an equal distance, thus the nearest neighboring points will be within $1/4$ voxel size. Then the space is partitioned into equal-spacing voxels. We choose the voxel resolution of one cubic meter per voxel, such a hyperparameter can be easily mod-

[1] Based on the statistics used in HoliCity [60], we sample arbitrary yaw and roll, and 0 deg -45 deg pitch.

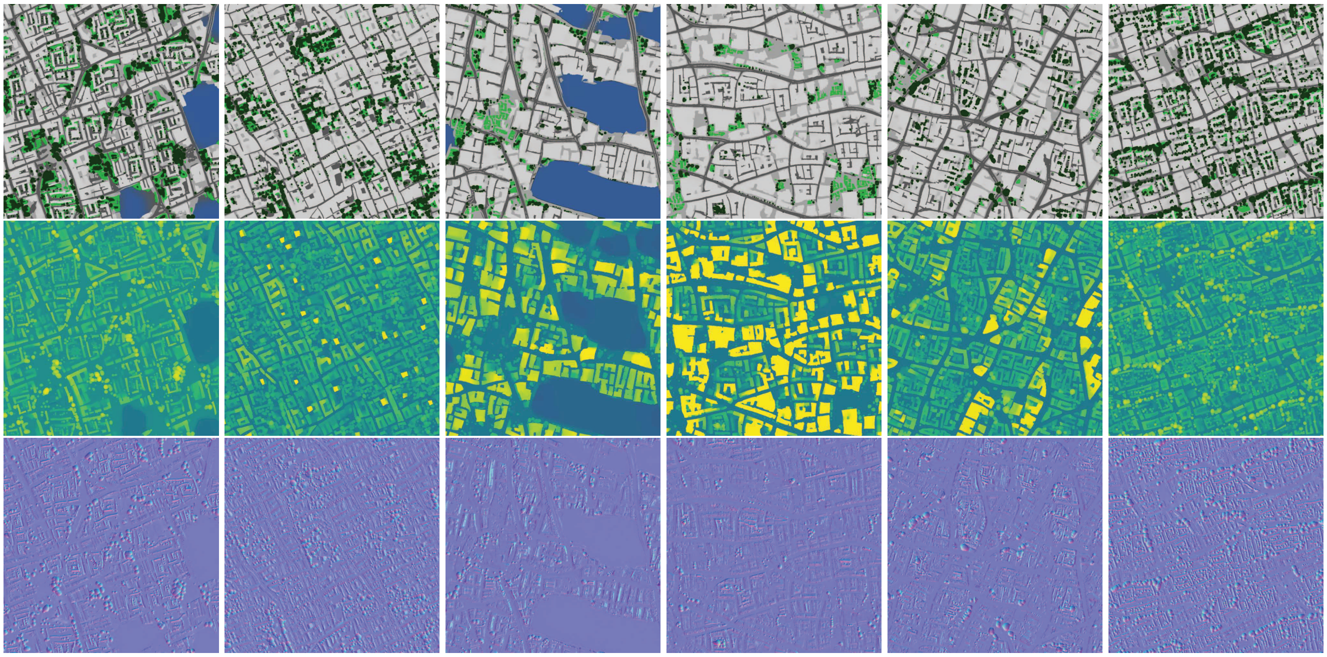

Figure 4: **Synthesized satellite maps.** We train InfinityGAN with contrastive discriminator in multiple data modalities (category, depth, and normal). The demonstrated images are 1024×1024 pixels cropped from the infinite-pixel images.

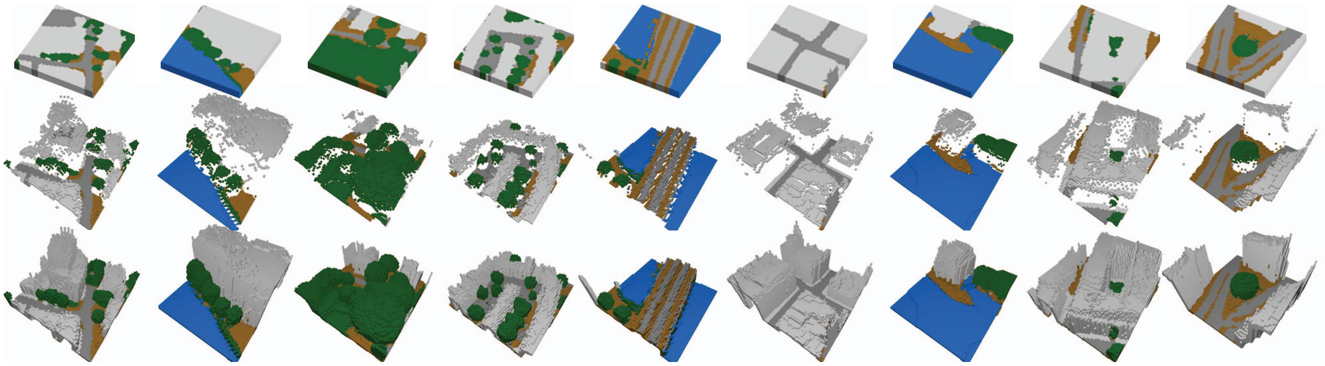

Figure 5: **Octree-based voxel completion.** High-quality and high-diversity voxels completed from synthetic satellite images. We show synthesized satellite images, lifted surface voxels, then 3D-completed voxels. The samples are 64^3 voxels.

ified to a higher resolution with sufficient memory [2]. Each voxel aggregates the point information with a majority vote for the category and vector averaging for the surface normal. The voxels are then partitioned and converted to a depth-6 octree, with 64 (=2^6) voxels on each edge. To further retrieve the satellite-image training data for the infinite-pixel satellite-view synthesis module, we scan the voxels from the top-down direction, retrieve the aggregated voxel information (i.e., category, depth, and normal) from the first-hit voxels, and project them onto a 2D image.

4.2. Infinite 2D Map Generation

Multi-modality map synthesis. In Figure 4, we show the satellite images synthesized with our method across the categorical, depth, and normal modalities. The images are 1024×1024 pixels cropped from the infinite-pixel images, which is equivalent to 1024×1024 square meters in the real world. The results show excellent structural alignments across modalities while maintaining high-quality and high-diversity appearances.

Ablate contrastive patch learning. In Table 1, we show that the contrastive patch discriminator is an important component in the pipeline that significantly improves the generator quality. Naively applying an additional patch discrimi-

[2] We run the InfinityGAN with 2× V100-16GB at batch size 32, OUNet with 1× V100-16GB at batch size 16, and GANcraft with 8× V100-32GB at batch size 8. Therefore, the memory consumption of InfiniCity training is mainly bounded by the neural rendering algorithm.

Table 1: **Contrastive patch learning improves InfinityGAN.** We train InfinityGAN at 197^2 pixels. The contrastive patch discriminator shows a significant performance boost.

Model	FID (↓)
InfinityGAN	115.6
InfinityGAN + patch discriminator	103.8
InfinityGAN + contrastive patch discriminator	**77.0**

Table 2: **Quantitative comparisons in structure synthesis.** InfiniCity outperforms the end-to-end synthesis method PVD [59], showing the efficacy of modeling structures from the satellite view.

Method	P-FID (↓)
PVD [59]	12.06
InfiniCity (Ours) w/ pillar completion	8.61
InfiniCity (Ours) w/o bilateral	6.92
InfiniCity (Ours)	**6.08**

Table 3: **Quantitative comparison with GSN**. InfiniCity substantially outperforms GSN in FID and KID (both the lower the better).

Method	FID (↓)	KID (↓)
GSN [12]	333.92	0.3252 ± 0.0017
Ours	**108.47**	**0.0842 ± 0.0008**

Table 4: **Paired Losses Impairs SPADE Generalization**. We train SPADE on HoliCity real data, then evaluate on InfiniCity synthetic worlds. The results show removing paired losses (feature matching and perceptual reconstruction losses) improves both performance and generalization.

Method	Eval Data	FID (↓)	KID (↓)
SPADE (original)	HoliCity	20.1	0.0135 ± 0.0003
SPADE w/o paired losses	HoliCity	**15.34**	**0.0067 ± 0.0003**
SPADE (original)	InfiniCity	31.38	0.0221 ± 0.0006
SPADE w/o paired losses	InfiniCity	**21.82**	**0.0135 ± 0.0004**

nator does not bring a similar level of performance gain.

4.3. Voxel World Completion

In Figure 5, we show the qualitative performance of our voxel completion model. The model ensures the final voxel structure is watertight and maintains the original voxel surfaces generated in the satellite-image synthesis step.

First, we would like to corroborate the significance of synthesizing structure from the satellite image. Despite not being easily scalable to an unconstrainedly large environment synthesis, it is intuitive to learn an end-to-end generator that directly synthesizes the 3D environment from scratch. We adopt PVD [59], a state-of-the-art point cloud generator, as a critical baseline. We convert the 64^3 voxels into point clouds using the center coordinate of the voxels. The model is trained end-to-end using 8,192 points per sample. Unlike the FID metric for images, there is no existing general and non-reference-based quality evaluation metric for 3D data. To measure the distribution distance similar to FID, we first pretrain an autoencoder using FoldingNet [54] on the real point cloud data and take the encoder as a feature extractor. Then we compute the feature distance in the FID manner. We denote the metric as P-FID. To compare with PVD, we also pick the center coordinate of the voxels to create point clouds from our synthesized voxels. As shown in Table 2, the proposed pipeline outperforms PVD even though the evaluation setting is in PVD's favor (i.e., it is trained and evaluated on point clouds).

Next, we justify the necessity of the 3D completion modules. A straightforward approach toward closing the gap between the surface and the ground is, for each surface voxel, to project the voxel to the ground level, and mark all voxels along the trajectory with the category of the original surface voxel. We call this naive baseline the "Pillar" method, as it essentially creates a pillar for each surface voxel. As an utterly simplified baseline method, such an approach can easily create undesired appearances for certain object classes, such as the trees. However, we show that even such an approach, heavily borrowing the structure constructed from the satellite images, can outperform PVD, the end-to-end diffusion-based point cloud synthesis baseline. Such results further show that synthesizing the structure from the satellite view can significantly simplify and benefit the structure synthesis procedure.

Finally, we ablate the usefulness of bilateral filtering. We utilize bilateral filtering to improve the plausibility of structure and suppress the noises from the synthesized satellite depth. In Table 2, we accordingly show that using bilateral filtering can further improve the P-FID.

4.4. Texturization via Neural Rendering

As we are the first attempt toward infinite-scale 3D environment generation jointly using 2D and 3D data, there is no existing baseline method for a fair comparison. Therefore, we compare with GSN [12], a model trained on trajectories of images with camera poses, to illustrate the advantages of making use of 3D data given existing techniques. In Figure 6 and Figure 7, we show the visual results of the images rendered on our synthesized-and-completed voxel world. The results show a robust 3D consistency of the 3D structure, as our underlying variables and rendering mechanisms are 3D grounded. In comparison, GSN [12] not only fails to learn the appearance of the city, but its latent space also fails to understand the 3D information due to lacking constrained and continuous camera trajectories to guide its formation. It only creates shape deformations and 2D trans-

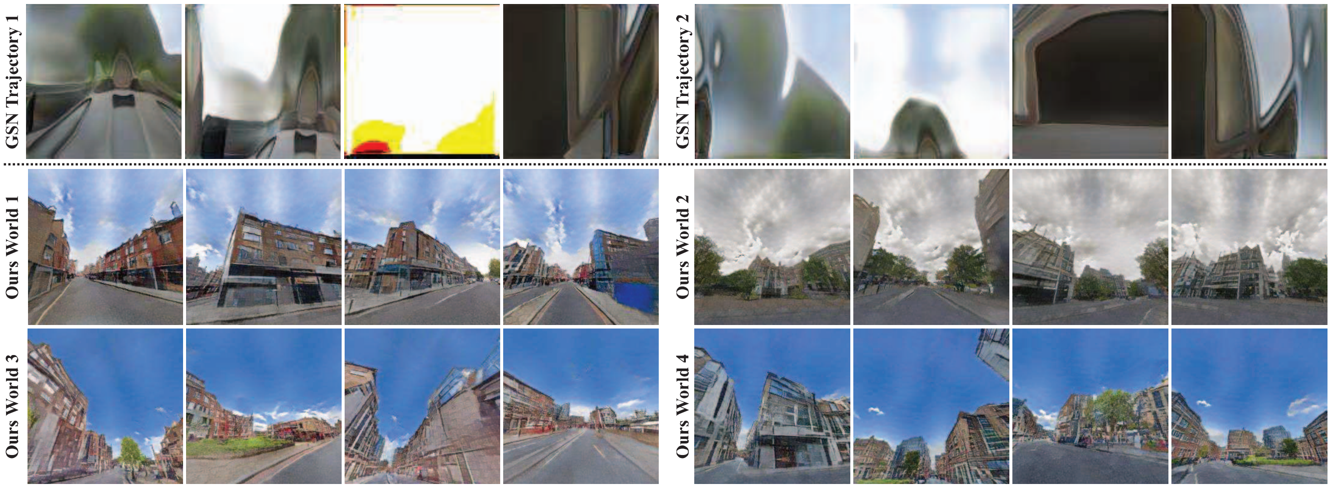

Figure 6: **Trajectory-wise image rendering results.** Our final rendering results present better quality, structural consistency, and diversity, over the competing method GSN [12]. Each group of images is rendered within the same voxel world using a shared global style code, while GSN shares the same global latent vector in each group.

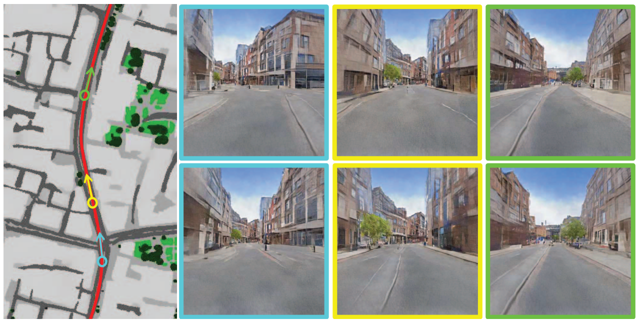

Figure 7: **Traversable and consistent 3D city rendering.** We render multiple views on a trajectory (red line) at three different locations (blue, yellow, and green) with small camera movements (marked as the arrow direction). The results show a strong cross-view consistency with a coherent style.

lations while traversing in its latent space.

Quantitatively, we further compare our method with GSN using FID [18] and KID [3] scores. For GSN, the images are randomly sampled from its latent space. For InfiniCity, we randomly sample the style latent vectors and *valid* (see Section 3.4) camera poses. We sample 2,048 images from both methods. The two sets of images are separately compared with a set of 2,048 street-view images randomly sampled from the HoliCity dataset. The quantitative evaluation shows InfiniCity substantially outperforms GSN in both FID and KID evaluations. Such a result further demonstrates our three-stage approach better captures the geometry and appearance of the city scene and is more suitable for modeling unconstrainedly large environments.

4.5. Impact of Misusing Paired Losses

The neural rendering stage requires pseudo-ground-truth images synthesized by a SPADE model pretrained on real data. SPADE model formulates its training task as a paired image-to-image translation. It leverages feature matching [50] and perceptual [57] losses between the real-synthetic image pairs to boost its training stability and synthesis quality. However, the reliance on supervised paired losses can limit the model's generalization to unseen data, such as the InfiniCity synthesized voxel worlds, and cause reduced visual quality in the final neural rendering. This domain gap is not only due to differences in the voxel distribution between real and synthetic domains, but also to the distribution of camera viewpoints. HoliCity uses street-view images from Google Maps, where the images are mostly captured by the camera mounted on top of a car. On the other hand, InfiniCity samples camera views from all *walkable* regions with a heuristic camera height (camera to voxel surface) sampled between 1.5 $\sim$ 2.5 meters.

In Table 4, we train the original SPADE and a variant without the paired losses on HoliCity real data. Then both models are tested on 50,000 segmentation images rendered on InfiniCity-generated worlds. We observe the model without paired losses shows better performance and generalization at the same time. While both models incur a level of degradation on synthetic data from InfiniCity, the degradation margin of the model without paired loss is substantially smaller.

5. Conclusion

In this work, we propose InfiniCity, a novel framework for unbounded 3D environment synthesis. We demonstrate that each stage of the framework produces high-quality and high-diversity results, and together create plausible, traversable, easily editable structures at an *infinite scale*.

With these exciting results, we observe that the quality of the final rendering is bounded by the neural rendering. As neural rendering is still at its early stage with rapid revolutions, many of the convergence and efficiency problems [43, 44, 55] are being recently addressed, and we expect that our neural rendering quality will substantially improve as our understanding of such a technique is deepened.

References

[1] Accucities. https://www.accucities.com/. 5

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 2

[3] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018. 8

[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. In *NIPS*, 2022. 2

[5] Lucy Chai, Richard Tucker, Zhengqi Li, Phillip Isola, and Noah Snavely. Persistent nature: A generative model of unbounded 3d worlds. In *CVPR*, 2023. 2

[6] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022. 2, 3

[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 *CVPR*, 2021. 3

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

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2

[10] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander Schwing, and Liangyan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *CVPR*, 2023. 2

[11] Zezhou Cheng, Menglei Chai, Jian Ren, Hsin-Ying Lee, Kyle Olszewski, Zeng Huang, Subhransu Maji, and Sergey Tulyakov. Cross-modal 3d shape generation and manipulation. In *ECCV*, 2022. 2

[12] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W Taylor, and Joshua M Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *ICCV*, 2021. 2, 3, 7, 8

[13] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 2

[14] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 2

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 3

[16] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. In *ICLR*, 2022. 2, 3

[17] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. Gancraft: Unsupervised 3d neural rendering of minecraft worlds. In *ICCV*, 2021. 2, 3, 4

[18] 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. In *NIPS*, 2017. 8

[19] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022. 2

[20] Jaebong Jeong, Janghun Jo, Sunghyun Cho, and Jaesik Park. 3d scene painting via semantic image synthesis. In *CVPR*, 2022. 3

[21] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020. 2, 3

[22] Zhengqi Li, Qianqian Wang, Noah Snavely, and Angjoo Kanazawa. Infinitenature-zero: Learning perpetual view generation of natural scenes from single images. In *ECCV*, 2022. 2

[23] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. Coco-gan: generation by parts via conditional coordinating. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4512–4521, 2019. 2

[24] Chieh Hubert Lin, Hsin-Ying Lee, Yen-Chi Cheng, Sergey Tulyakov, and Ming-Hsuan Yang. InfinityGAN: Towards infinite-pixel image synthesis. In *ICLR*, 2022. 2, 4

[25] 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 *ICCV*, 2021. 2

[26] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *CVPR*, 2021. 2

[27] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *ECCV*, 2020. 2

[28] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, 2018. 5

[29] 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 *ECCV*, 2020. 3

[30] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *CVPR*, 2022. 2

[31] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 2, 4

[32] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A. Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. In *NIPS*, 2020. 4

[33] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017. 3

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2

[35] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, 2020. 3

[36] Yuan Shen, Wei-Chiu Ma, and Shenlong Wang. Sgam: Building a virtual 3d world through simultaneous generation and mapping. 2022. 2

[37] Aliaksandr Siarohin, Willi Menapace, Ivan Skorokhodov, Kyle Olszewski, Jian Ren, Hsin-Ying Lee, Menglei Chai, and Sergey Tulyakov. Unsupervised volumetric animation. In *CVPR*, 2023. 3

[38] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces. In *ECCV*, 2022. 3

[39] Ivan Skorokhodov, Aliaksandr Siarohin, Yinghao Xu, Jian Ren, Hsin-Ying Lee, Peter Wonka, and Sergey Tulyakov. 3d generation on imagenet. In *ICLR*, 2023. 3

[40] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. In *ICCV*, 2021. 2

[41] Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. Epigraf: Rethinking training of 3d gans. In *NeurIPS*, 2022. 3

[42] Edward J Smith and David Meger. Improved adversarial systems for 3d object generation and reconstruction. In *CoRL*, 2017. 2

[43] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 9

[44] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *CVPR*, 2021. 9

[45] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998. 4

[46] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM TOG (Proc. SIGGRAPH)*, 2017. 3

[47] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM TOG (Proc. SIGGRAPH)*, 2017. 4

[48] Peng-Shuai Wang, Yang Liu, and Xin Tong. Deep octree-based cnns with output-guided skip connections for 3d shape and scene completion. In *CVPR Workshops*, 2020. 3, 4

[49] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive o-cnn: A patch-based deep representation of 3d shapes. *ACM TOG (Proc. SIGGRAPH)*, 2018. 3

[50] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 8

[51] Chenfei Wu, Jian Liang, Xiaowei Hu, Zhe Gan, Jianfeng Wang, Lijuan Wang, Zicheng Liu, Yuejian Fang, and Nan Duan. Nuwa-infinity: Autoregressive over autoregressive generation for infinite visual synthesis. In *NeurIPS*, 2022. 2

[52] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. In *CVPR*, 2018. 2

[53] Yinghao Xu, Menglei Chai, Zifan Shi, Sida Peng, Ivan Skorokhodov, Aliaksandr Siarohin, Ceyuan Yang, Yujun Shen, Hsin-Ying Lee, Bolei Zhou, et al. Discoscene: Spatially disentangled generative radiance fields for controllable 3d-aware scene synthesis. In *CVPR*, 2023. 3

[54] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 7

[55] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 9

[56] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2017. 4

[57] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 8

[58] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *ICML*, 2017. 4

[59] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *ICCV*, 2021. 7

[60] Yichao Zhou, Jingwei Huang, Xili Dai, Linjie Luo, Zhili Chen, and Yi Ma. HoliCity: A city-scale data platform for learning holistic 3D structures. *arXiv preprint arXiv:2008.03286*, 2020. 2, 5