SynCity 3000
Bootstrapping Scene-Scale 3D Diffusion

Paul Engstler, Iro Laina, Christian Rupprecht, and Andrea Vedaldi

Visual Geometry Group, University of Oxford
{paule,iro,chrisr,vedaldi}@robots.ox.ac.uk
https://research.paulengstler.com/syncity-3k

Abstract. We present SynCity 3000, a framework for generating 3D scenes that are globally coherent while enabling fine-grained layout control. Building on the ability of current image-to-3D generators to produce complex 3D assets from a single image, we extend this capability to the scale of entire scenes by adapting the generator to be applicable as a convolutional operator. We achieve this by fine-tuning the model on scene-like data generated by a new synthetic data engine, which we propose to address the scarcity of 3D scene data for training. The convolutional generator is then applied to a dimetric image of the entire scene, generated from the user prompt, resulting in 3D scenes of arbitrary size and complexity. Across diverse prompts and layouts, SynCity 3000 produces large, coherent, and detailed scenes, addressing the shortcomings of prior approaches to 3D scene generation.

Keywords: 3D scene generation

1 Introduction

Creating 3D content for movies, games and simulations is a time-consuming and labor-intensive task that requires skilled artists and designers. Recent models that can produce automatically high-quality 3D assets from text prompts [36, 56, 62, 79], but these are usually limited to single objects. Generating entire 3D scenes would be much more impactful in applications. SynCity [15] has recently demonstrated that off-the-shelf models for image generation and image-based 3D reconstruction of single objects can be repurposed to generate large scenes. By building on off-the-shelf models, SynCity sidesteps the lack of large and diverse 3D scene datasets for training a corresponding generator from scratch. It achieves this by reinterpreting the scene as a grid of tiles, each of which is akin to an object which can be generated semi-independently. While this works, the grid-like structure is clearly visible in the final output.

In this paper, we address this limitation by introducing SynCity 3000, a two-stage framework for generating large-scale 3D scenes from text prompts (Figs. 1 and 3). Breaking free from the fixed grid-like structure of SynCity, our approach creates 3D worlds of arbitrary structure and complexity. In the first stage, we generate a 2D image template that defines the appearance and layout of

Fig. 1: Diverse 3D worlds of arbitrary size and complexity are easily created with SynCity 3000 from scratch. Our approach first constructs a visually and semantically coherent 2D template of the entire scene, and then converts it into 3D Gaussian Splats with a fine-tuned two-stage generative diffusion model.

the scene. In the second stage, this template is converted into the final 3D scene. This pipeline operates automatically without manual intervention. Notably, the two stages are independent, allowing any template-like image, including those created by graphic artists, to be used for generating a 3D scene.

The stages require repurposing off-the-shelf models for 2D and 3D generation. This necessitates several innovations, which we summarize next and in Sec. 3.

In the first stage, we prompt an image generator to create a high-resolution 2D template of the scene. The generator is based on latent diffusion and, in order to generate scenes of any extent, we divide the 2D latent space into partially overlapping windows without forcing them to look like square tiles. Optional layout constraints can be introduced to allow fine-grained control of the generated scene. By conditioning on all constraints and averaging their contributions at each step of the image diffusion process (inspired by MultiDiffusion [5]), windows blend in naturally with their surroundings.

In the second stage, we use an image-conditioned 3D generation model to transform the template into a 3D scene. We employ a voxel-based approach, where a coarse, voxelized version of the world is generated first and then enriched with rich visual and semantic features for each voxel. These are then decoded into highly detailed scenes represented by 3D Gaussian Splats. Similar to the template generation, we subdivide the scene (or the corresponding latent code) into partially overlapping windows to support scenes of arbitrary size. We fine-tune the 3D generator to operate on the sliding window in a "convolutional" manner, and also introduce additional adaptation that facilitates learning this new behaviour. Fine-tuning uses synthetic data we create procedurally for this purpose, as detailed in Sec. 4.

Finally, in Sec. 5, we demonstrate the effectiveness of SynCity 3000 on a variety of scenes of different sizes and complexities. Empirically, we show that our approach outperforms SynCity and other reconstruction methods both qualitatively and quantitatively (Figs. 6 and 7 and Tab. 1).

2 Related work

Image-based scene generation. The ability of 2D generative models to generate increasingly realistic images while exhibiting a semantic and geometric understanding of the depicted scene [9,73] has inspired a wide range of works [10, 16,21,45,47,51,59,69,70,75] that utilize them to create representations of large-scale scenes in the form of meshes, Neural Radiance Fields [42], or 3D Gaussian Splats [27]. Most of these methods leverage monocular depth estimation models [4,7,26,64] to project images and apply various heuristics to fuse them in space. Other works directly operate on panoramas [34,52,57,60] or posed images [29]. However, the generated scenes frequently exhibit geometric artifacts and holes in the scene due to occlusions when steering away from the viewpoints used to generate the scene.

Asset-based scene generation. By populating a scene with 3D assets based on a synthesized layout, geometric and occlusion issues are easily resolved. In this line of work, layouts are either directly based on an image [2,13,19,20,23,30,67], or generated with large language models [17,22,43,53,66,82] or trained diffusion models [39,54,71]. Assets may be retrieved from a library or generated on-the-fly, enabled by object-centric 3D generation models [35,38,62,78]. The generated scenes, however, are frequently monotonous, at times unrealistic, suffer from inter-object collisions, and are largely restricted to indoor scenes.

Explicit scene generation. Geometric consistency issues can be circumvented by directly synthesizing the scene representation, as presented in more recent works. LT3SD [41] proposes to learn a diffusion model, which employs a patch-by-patch and coarse-to-fine approach to build 3D environments without conditioning. $\mathcal{X}^3$ [49] can generate large-scale 3D scenes by representing them as a hierarchy of sparse voxel grids. Both require large-scale, domain-specific training data, and are either incapable or struggle with the conditioning of their generation process. BlockFusion [61] learns a diffusion model to extend a mesh auto-regressively by small blocks, SceneCraft [65] one to generate scene renderings. While the layout of the generated scene can be controlled, these methods require the availability of domain-specific datasets. Recent work [32,33] generates large-scale scenes but focuses on semantic categories and thus on semantic completion of an existing scene rather than providing text or image interfaces.

In SynCity [15], a scene is reinterpreted as a grid of tiles that are generated individually and then fused together. This process is steered by prompts on a global and tile level. 3D tiles are generated sequentially in a training-free approach with TRELLIS [62], leveraging its ability to create high-quality 3D objects from large-scale 3D object-centric datasets [11,14,18,28]. While the generated scenes follow an overall theme, the sequential tile generation pattern causes scenes to lack overall coherence, leading to a busy, grid-like appearance.

Another training-free approach based on TRELLIS [62], 3DTown [80], transforms an image of a scene into 3D Gaussian Splats. Here, a monocular depth estimator [58] is used to build a geometric prior of the scene. It is then partitioned into overlapping regions, locking parts already observed through the

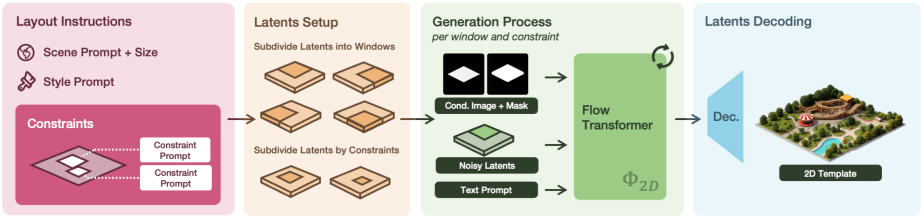

Fig. 2: 2D Generation Pipeline. We generate a dimetric scene template based on text prompts and optional layout constraints. After subdividing the latents into windows, we apply a latent diffusion model to jointly denoise the latent canvas. Once complete, the latent canvas is then decoded into the final 2D scene template.

image. The unknown parts are then completed by adding a mask in the 3D generation process. While the generated scenes follow the input image, they suffer from geometric artifacts, holes, and limited resolution.

In NuiScene [31], a conditional outpainting diffusion model is trained on a subset of Objaverse [14], which contains 43 scenes. Scenes are produced in chunks, based on a learned VecSet [74] encoding, enabling unbounded generation. However, there are visible discontinuities between chunks, textures have to be generated with SceneTex [8], and the small dataset size severely limits the method's ability to generalize.

In this work, we address the limitations of these prior works. We propose a simple method inspired by MultiDiffusion [6] (and similar to GLIGEN [37]) to create an arbitrarily complex scene template, which strictly generalizes SynCity. To be able to turn this template into a 3D representation, we fine-tune a 3D generation model [62] to support convolutional inference, thus tailoring it to the task rather than building brittle heuristics to use an object-centric model, as in 3DTown. This fine-tuning is enabled by our proposed synthetic dataset engine, which generates scene-like data, allowing us to evade the lack of large-scale 3D scene datasets that are not strictly limited to indoor settings [3,12,28,46,48,50,68]. This engine allows us to scale far beyond the training dataset of NuiScene.

3 Method

The input to SynCity 3000 is a high-level textual description of the scene which consists of the theme prompt P, which describes the overall theme of the scene (e.g., "a bustling medieval town square"), and the style prompt P_{style}, which describes its visual appearance (e.g., "a bright sunny day"). Optionally, the method can also take as input a set of *local prompts* P_w (e.g., "a stone well") each associated with a *window* $w \in \mathcal{W}_c$, further defined below.

The output of our method is a full 3D scene represented as 3D Gaussian Splats [62], which can be rendered from arbitrary viewpoints.

As illustrated in Fig. 2 and Fig. 3, SynCity 3000 comprises two independent stages: the first stage (Sec. 3.1) generates a 2D template of the scene and the second stage (Sec. 3.2) generates a corresponding 3D model.

3.1 2D template generation

Stage 1 generates a 2D image *template* J that defines the visual appearance and layout of the scene as a whole. As in SynCity, this stage repurposes a pre-trained text-to-image model Φ_{2D} that uses latent denoising diffusion. Although off-the-shelf, the model does not directly generate images suitable for 3D reconstruction. We develop a prompting strategy to enable the model to generate *dimetric views*. We follow SynCity and run the generator in "inpainting mode", passing to it a "base image" I_{base} that shows a dimetric perspective of a supporting structure (similar to a concrete slab), and an inpainting mask M_{base} that covers the area above this base, as shown in Fig. 2. The inpainted image output by Φ_{2D} then has the required dimetric framing.

A limitation of the scheme above is that the extent of the scene is bound by the maximum resolution of the generator Φ_{2D}. SynCity addressed this issue by generating and combining non-overlapping scene tiles, one at a time. While this process can be extended indefinitely, it is prone to visual artifacts when tiles are reconstructed in 3D and assembled. To address this issue, we propose a new approach that results in a much better and more coherent global structure of the scene. Inspired by MultiDiffusion [5], we apply Φ_{2D} to overlapping windows *throughout* the denoising process. Thus, information is passed across windows at every denoising step, resulting in a globally coherent image.

In more detail, the denoising model Φ_{2D} operates in *latent space*, given by a grid of tokens $\boldsymbol{z} \in \mathbb{R}^{C\times H/\sigma\times H/\sigma}$ where $H \times H$ is the resolution of the corresponding 2D image, σ is the token downsampling factor, and C is the latent channel dimension. We assume for simplicity that all images are square, and we call $\boldsymbol{z}$ the *latent canvas*.

We further assume that the model Φ_{2D} operates on smaller token grids $\boldsymbol{z}_w \in \mathbb{R}^{C\times S/\sigma\times S/\sigma}$, where $S \leq H$ is the size of the image window that can be generated by the off-the-shelf model. Hence, the network Φ_{2D} can be seen as a function $\boldsymbol{\epsilon}_w = \Phi_{\text{2D}}(\boldsymbol{z}_w | I_{\text{base}}, M_{\text{base}}, P_w, P_{\text{style}})$ estimating the noise $\boldsymbol{\epsilon}_w$ in the code $\boldsymbol{z}_w$. Each application of Φ_{2D} is conditioned by the global style prompt P_{style} as well as the local content prompt P_w, which is window-dependent. In particular, $P_w = P$ if $w \notin \mathcal{W}_c$, i.e., if the window is not associated with a local prompt.

The windows $\mathcal{W}$ are thus squares of size S sliding over the generated image with a stride $s < S$ to ensure overlap, with the addition of any windows defined by layout constraints $\mathcal{W}_c$ (so that $\mathcal{W}_c \subset \mathcal{W}$). The symbols $\boldsymbol{z}_w$ denote the corresponding subset of the latent canvas $\boldsymbol{z}$ covered by the window w. At each denoising step, the *entire* canvas $\boldsymbol{z}$ is updated after averaging the estimated noise $\boldsymbol{\epsilon}_w$ over all windows $w \in \mathcal{W}$, similar to MultiDiffusion [5].

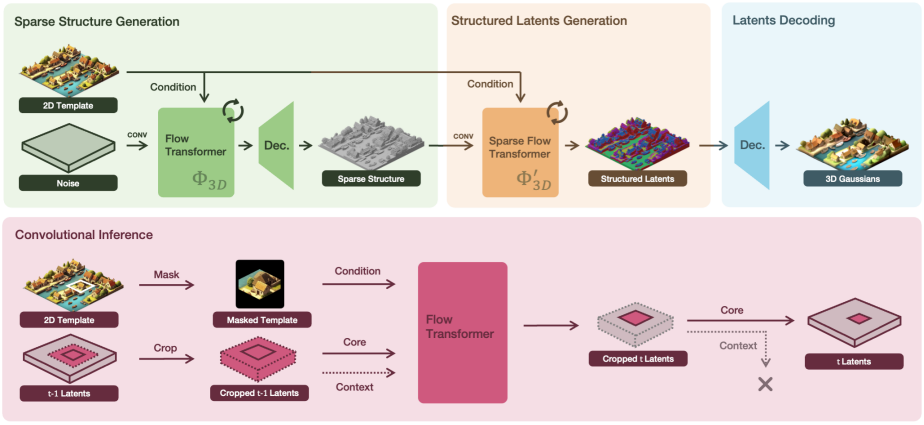

Fig. 3: 3D Generation Pipeline. We use a multi-stage pipeline to synthesize full 3D scenes from 2D templates using convolutional inference. In this inference process, we divide the latent into smaller regions that we jointly denoise. Each region has a direct template correspondence. To transform a scene, we first synthesize the sparse structure—a coarse voxel grid. Then, we infer features that encode the appearance and semantics of each voxel. Finally, we decode these structured latents into the final 3D Gaussian Splats representation. We use the decoders of [62].

3.2 3D generation

We now turn to the task of transforming the 2D template J obtained in stage 1 (Sec. 3.1) into a 3D scene. Similar to 2D generation, SynCity demonstrated that off-the-shelf image-to-3D models like TRELLIS [62] can be repurposed for this task. In SynCity, the idea is to reconstruct the 3D scene tile-by-tile. This works because individual tiles have a complexity that roughly matches a single 3D object and can be reconstructed relatively well by TRELLIS without any further training. However, there are artifacts where the resulting 3D tiles join up. Furthermore, it requires tiles to be represented as separated images, which leads to visual and semantic discontinuities as discussed earlier.

As our stage 1 outputs a single, large image of the complete scene in its entirety, we fine-tune TRELLIS to operate in a *convolutional* manner. This allows us to process the entire scene at once, potentially for an arbitrarily large size of the template image J.

TRELLIS itself comprises two generators: the first determines the *sparse structure* of the object, which is a coarse voxel grid outlining its geometry, and the second samples *structured latents*, which are features attached to each voxel that encode shape and appearance details. Finally, the structured latents are decoded into 3D Gaussian Splats. We begin by discussing how the first generator can be made convolutional (Sec. 3.2), and then we move to the second one (Sec. 3.2).

Convolutional sparse structure generation. Given an image I of the object, the first of the two generators in TRELLIS is tasked with sampling a corre-

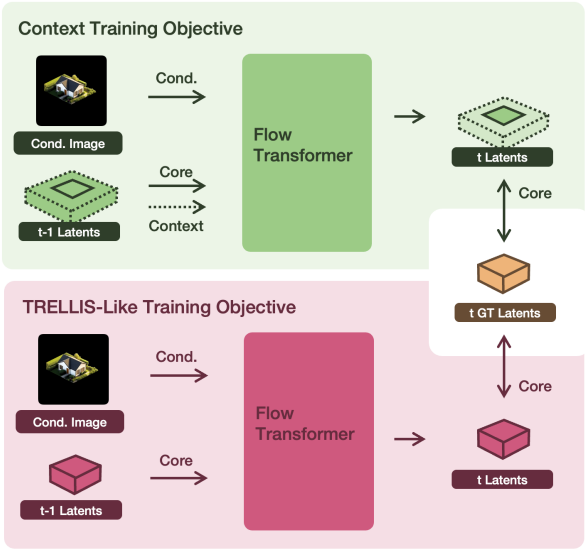

Fig. 4: Fine-Tuning Design. We utilize two objectives in our fine-tuning process. In the first (top), we provide the core *and* surrounding context to the model. In the second (bottom), we only provide the core, which imitates the original objective of TRELLIS [62]. In both cases, we use a masked conditioning image and apply the loss only to the core.

sponding voxel grid $\mathcal{O} \in \{0,1\}^{N\times N\times N}$ which captures its rough shape. TRELLIS auto-encodes $\mathcal{O}$ to a lower-resolution latent grid $\boldsymbol{o} \in \mathbb{R}^{M\times M\times M\times C}$, where $M \ll N$ and C is the latent channel dimension.

To reconstruct $\boldsymbol{o}$ from an image I, TRELLIS uses a denoising Diffusion Transformer (DiT) Φ_{3D}. Let $\boldsymbol{p} \in \{0,\dots,M-1\}^{M\times M\times M\times 3}$ be the positional indices of the latent vectors in $\boldsymbol{o}$. The latent grid $\boldsymbol{o}$ is reinterpreted as a tensor $M^3\times C$ of M^3 tokens, tokens are summed with sinusoidal positional encodings $\rho(\boldsymbol{p}) \in \mathbb{R}^{M^3\times C}$ representing their position in the grid, and the tokens and image are fed to the transformer to obtain the denoising signal $\boldsymbol{\epsilon} = \Phi_{\text{3D}}(\boldsymbol{o} + \rho(\boldsymbol{p})|I)$.

Our goal is to modify this scheme to reconstruct much wider scenes, i.e., $\mathcal{O} \in \{0,1\}^{N_{\text{scene}}\times N_{\text{scene}}\times N}$ with $N_{\text{scene}} \gg N$. The auto-encoder is already convolutional, so, applied to a larger occupancy grid, it produces a correspondingly larger latent grid $\boldsymbol{o} \in \mathbb{R}^{M_{\text{scene}}\times M_{\text{scene}}\times M\times C}$ with $M_{\text{scene}} \gg M$. The denoiser Φ_{3D}, however, is not convolutional, so it needs to be modified to be applicable to the new setting.

To apply Φ_{3D} to this larger latent grid, we consider overlapping sliding windows $w \in \mathcal{W}$. Each window extracts a sub-grid $\boldsymbol{o}_w$ of size $M \times M \times M$ from the larger latent grid $\boldsymbol{o}$, making it possible to apply the model. The conditioning signal must also be modified to indicate which portion of the world is encoded by $\boldsymbol{o}_w$ and thus needs to be reconstructed. We do so by considering a corre-

sponding image crop I_w centered on the same portion of the world as $\boldsymbol{o}_w$. Thus, the denoising model is $\boldsymbol{\epsilon}_w = \Phi_{\text{3D}}(\boldsymbol{o}_w + \rho(\boldsymbol{p})|I_w)$, where the positional encoding $\boldsymbol{p}$ remains unchanged, as it is relative to the window rather than the world. All windows are processed in parallel at each denoising step, and the resulting denoising signals $\boldsymbol{\epsilon}_w$ are averaged to update the entire latent grid $\boldsymbol{o}$.

Note that the statistics of $(\boldsymbol{o}, I)$ in the original model are fairly different from those of $(\boldsymbol{o}_w, I_w)$ in the convolutional setting above. For one thing, $\boldsymbol{o}_w$ only encodes a well-defined portion of the scene, which we call the *core*, but the image I_w shows not only this portion, but also the area around it, which should not be encoded by $\boldsymbol{o}_w$. We do three things to bridge this gap.

First, as further detailed below, we fine-tune the model Φ_{3D} to operate in this new setting, utilizing the training data we develop in Sec. 4.

Second, we focus the model on the core by masking out the part of the image I_w that *does not* correspond to it, passing to it $I_w \odot M$, where M is the mask. See Fig. 3 for an illustration.

Thirdly, we provide further 3D context to the model by expanding the latent grid $\boldsymbol{o}_w$ to include voxels surrounding the core, which results in additional contextual tokens $\boldsymbol{o}_w^c$ that, in combination with the core, cover an area M_{context} larger than the core alone (so overall $M < M_{\text{context}} \ll M_{\text{scene}}$). This helps each application of the model to update the core while accounting for the 3D geometry of the surrounding area. This also requires assigning positional encodings $\boldsymbol{p}^c$ to the context tokens, which we do carefully as explained below. Hence, the denoising model becomes

$$\boldsymbol{\epsilon}_w = \Phi_{\text{3D}}(\boldsymbol{o}_w + \rho(\boldsymbol{p}), \boldsymbol{o}_w^c + \rho(\boldsymbol{p}^c)|I_w \odot M).$$

Positional encoding for the context tokens. Before, we noted that the core tokens $\boldsymbol{o}_w$ maintain the same positional encoding $\rho(\boldsymbol{p})$, which are relative to the window, as in the original model. When no context tokens are provided, this allows the model to operate in a similar manner as the original one as much as possible. However, when context tokens $\boldsymbol{o}_w^c$ are provided, we must assign positional encodings $\rho(\boldsymbol{p}^c)$ to them too. The natural choice is to define them as positions outside the core.

In other words, given the original positional embedding function $\rho : \{0, \dots, M-1\}^3 \mapsto [-1, 1]^C$, we extend it to the larger domain $\{-V, \dots, M+V-1\}^3$ where $M_{\text{context}} = M + 2V$ is the size of the core plus context. By constructing the grid such that $\boldsymbol{p}^c$ span from $(-V, -V, 0)$ to $(M+V-1, M+V-1, M-1)$, the core receives the original positional embeddings.

Fine-tuning scheme. Besides the modifications above, we fine-tune TRELLIS to teach it to operate convolutionally. However, we also wish to keep its ability to generate high-quality 3D objects and prevent catastrophic forgetting.

Thus, we use two separate training objectives to achieve both goals. Their occurrence is determined by a predefined probability p. We either provide the core *with* or *without* context. Thanks to our design choices, the latter then mirrors the original task of TRELLIS (see Fig. 4). The conditioning image is the same in both cases. We use a mean-squared error loss that is only applied to the

core. This reinforces the notion that the models need to consider the context, but only responses within the core are relevant.

Convolutional structured latents generation. Once the sparse structure $\mathcal{O}$ has been generated, TRELLIS associates to each non-zero voxel at a given position $\boldsymbol{p}_i$ a corresponding feature vector $\boldsymbol{z}_i$ encoding local appearance and shape. The resulting set of $L = \|\mathcal{O}\|_1$ *structured latents* is stacked in a tensor $\boldsymbol{z} \in \mathbb{R}^{L \times C}$, with corresponding positional indices $\boldsymbol{p} \in \{0, \dots, N-1\}^{L \times 3}$. These are then passed to one more denoising diffusion model $\boldsymbol{\epsilon} = \Phi'_{\text{3D}}(\boldsymbol{z} + \rho(\boldsymbol{p})|I)$ to sample $\boldsymbol{z}$ given the image I. Except for the fact that the tokens $\boldsymbol{z}$ are sparse instead of dense, this model is very similar to the one described in Sec. 3.2, so we can apply the same convolutional scheme.

Scene inference. With the models in place, we can now directly use them to transform a template into a complete scene represented by 3D Gaussian Splats, as illustrated in Fig. 3.

Similar to how we set the size of the canvas during the 2D template generation (see Sec. 3.1), we also set a latent resolution for the 3D generation $\mathcal{R}_{\text{latents}}$ in integer multiples of the core resolution (i.e., $\mathcal{R}_{\text{latents}} \mod N = 0$). This establishes a clear correspondence between pixel (template) and latent space. Crucially, we are free to choose these values independently, and thus directly control the size and level of detail of the generated scene.

Both the sparse structure and structured latents generation rely on convolutional inference that we have enabled through our design choices and training scheme. We divide the template and latents into windows using strides s_{pixel} and s_{latents}. Then, we isolate that particular window by masking the template and cropping the latents. The latents contain the core, which corresponds to the part of the world shown in the masked template, and its surrounding context. While the models denoise the latents of the core and context, we only retain its predictions for the core. At each denoising timestep, we denoise all windows, averaging their contributions if a small stride results in overlaps.

Finally, once we have obtained the complete scene as Gaussians, we apply a simple color correction to address a shortcoming of the TRELLIS model, which we detail in the supplementary materials.

4 Dataset

While a growing body of large-scale 3D datasets for objects [11,14,18,28] exists, the collection of scene-scale data has been lagging behind. Data availability has been a key factor in improving the quality of 3D generative models [62,76,78]. As we aim to fine-tune the models of TRELLIS to operate on large-scale scenes, the lack of 3D scene data poses a problem.

Therefore, we introduce a synthetic dataset engine, which generates complete 3D scenes in a manner that harmonizes with our scene-scale diffusion scheme,

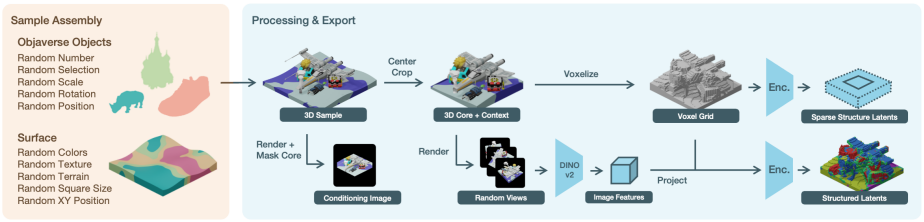

Fig. 5: Data Generation. We propose a dataset engine to generate diverse 3D scenes by placing random Objaverse-XL [14] objects on randomized terrains. Each scene is rendered in dimetric projection to match the 2D template generation Fig. 2. The rendering of the scene is masked to only show the core, whereas we crop the 3D scene to include the core and some context. This 3D crop is then converted into a voxel grid and infused with DINOv2 [44] features. We then encode it into sparse structure and structured latents using the encoders of [62].

as in Sec. 3.2. In the spirit of LRM-Zero [63], which uses random assortments of shapes and textures to bootstrap a 3D generator, we place Objaverse-XL objects onto randomly generated surfaces. For an overview of our proposed generation process, see Fig. 5.

Sample assembly. To mirror the way we generate our 2D templates (see Sec. 3.1), we first create a surface of variable size with random terrain. We blend multiple random colors to texture it, with hard-stop transitions between colors. Then, we pick a random number of Objaverse objects and randomly place them on top of the surface. Objects are randomly scaled and placed in such a manner that they do not sit too close to the surface's margin. We illuminate the scene evenly from the top, and add a overhead light source to cast shadows randomly. Due to the highly stochastic nature of this process, it yields a large variety of samples, imposing very few restrictions on the content to reconstruct.

Processing & export. We use an orthographic camera to render the scene in dimetric projection, mirroring the templates we generate (see Sec. 3.1). We consider the unit cube, which surrounds the world origin, the *core*. To obtain the matching voxel grid, which serves as the training target, we crop a cube with a resolution of $M_{\text{context}} \times M_{\text{context}} \times M$ voxels around the origin (recall from Sec. 3.2). This crop contains the core at its center (up to M^3 voxels) as well as additional voxels for context. Following [62], we render $\mathcal{V}_{\mathcal{F}}$ additional views using a perspective camera, extract DINOv2 [44] features, and project them onto the voxel grid. This yields a sparse structure and its corresponding structured latents.

Technical details. We implement the data engine using Blender. We choose $\mathcal{V}_{\mathcal{F}} := 32 + 5$, where we generate 32 random views of the scene from the top, a view from each side, as well as a view from the bottom.

Table 1: Template faithfulness. We evaluate multiple metrics to gauge the faithfulness of 3D scene reconstructions to the template with LPIPS [77], structural similarity index measure (SSIM), and peak signal-to-noise ratio (PSNR). Our method with all proposed design choices leads to superior performance.

Method	LPIPS ↓	SSIM ↑	PSNR ↑
TRELLIS [62]	0.4094	0.4966	13.5911
TripoSG [35]	0.4182	0.4953	12.2768
Hunyuan3D-2.1 [24]	0.4689	0.4490	11.5394
Ours (no fine-tuning)	0.4726	0.4657	11.7622
Ours (no context)	0.4121	0.5142	14.2038
Ours (smaller context)	0.4000	0.5047	14.1266
Ours (large stride)	0.4143	0.5192	14.0624
Ours	**0.3993**	**0.5247**	**14.4137**

5 Experiments

Experimental details. We use our dataset generation engine to produce 320k samples. For fine-tuning the sparse structure and structured latent transformers of TRELLIS [62], we use a learning rate of 5×10^{-6}, not freezing any layers, and a batch size of 1 (due to the additional voxels introduced). The sparse structure model was fine-tuned for 260k steps, the structured latent model 660k steps on 2× NVIDIA RTX A6000 GPUs.2 The sparse structure model has a resolution of $(M_{\text{context}})^2 \times M = (M + 2V)^2 \times M = (16 + 2 \cdot 8)^2 \times 16$, the structured latent model $(N_{\text{context}})^2 \times N = (N+2V)^2 \times N = (64+2 \cdot 32)^2 \times 64$. The training objective task probability is set to $p = 0.5$. The 2D inpainting model used during template generation is the Flux ControlNet of [1]. In the following, we present scenes that were obtained with layout instructions entirely generated by the large language model (LLM) ChatGPT 5 Instant. We choose a template window width of 896 px (in the 2 : 1 dimetric projection, the height is thus 448 px), which corresponds to M during the first stage and N in the second stage. Put differently, this is the size of a single core patch in the template. Overall, the scene templates measure 1344×672 px. The masks of the template windows are extruded by 60 px. This imposes a maximum height of tall structures at the boundaries that we reconstruct. This value can be freely chosen but needs to be consistent in dataset generation, training, and scene inference. During the convolutional inference, we use a stride of 0.5 patches.

Comparisons. We compare SynCity 3000 across multiple axes: layout control, faithfulness to the template, scene-scale reconstruction quality, and overall scene quality. To this end, we conducted a user study (Table 2) and present multiple quantitative and qualitative results.

Unanimously, users prefer our flexible layout control over that of SynCity, which only allows rather rigid grids with a low resolution, severely limiting the

Fig. 6: Comparison With SynCity [15]. We compare scenes generated by SynCity (left) and our method (right) using the same prompt for each. Overall, the worlds generated by our method are richer, more vibrant, and more coherent, both visually and semantically. They have an organic structure that lets elements flow across the whole scene.

creative freedom. This becomes apparent in Fig. 6, where we show qualitative comparisons to SynCity. We query the same LLM to produce instructions with the same theme prompt. SynCity creates scenes with a clearly apparent grid structure due to its tile-based generation mechanism. While this approach works for scenes where this setting is natural (*e.g.*, grid-planned cities), it falls short for those that are more open or have larger structures that span beyond a single tile (see the bottom row in Fig. 6). This is a methodological limitation of SynCity, which is fully alleviated by our convolutional approach.

A key part of our method is accurately translating the generated 2D template into an actual 3D scene. Thus, we evaluate the faithfulness of the reconstructed scene to the templates next, comparing it with off-the-shelf image-conditioned models TRELLIS [62], TripoSG [35], and Hunyuan3D-2.1 [24], as presented in Tab. 1. We use a set of 35 templates, which were randomly generated using an LLM. Based on this set, we generate scenes using all methods and render the results with an orthographic perspective that matches the template. Then, we use image similarity metrics to evaluate the faithfulness of the reconstructions to their corresponding image templates. We find that object-centric methods produce scenes at resolutions insufficient for this scale and thus lack finer details and occasionally deviate significantly from the conditioning image. This becomes especially noticeable for larger scenes, as presented in Figure 7. Please refer to the supplementary materials for additional qualitative comparisons.

The geometric quality of the reconstruction, apart from mere faithfulness to the template, also matters. While scene generation is an inherently generative task with no ground-truth data, we can leverage our synthetic dataset engine to obtain scene-like proxies. These allow us to gauge the geometric accuracy of the scene reconstructions as if ground truth were available. To consider the effects of scene scale, we produce 25 scenes at two different scales, leading to different template sizes when rendered (one as used throughout this section, and one at $5 \div 3$ of that size). For evaluating LPIPS and PSNR, we use 32 random views. We report the results in Table 3. While the performance of TRELLIS deteriorates as the scene size increases, ours stays consistent and ahead in most metrics.

Table 2: User preference evaluation. We asked users ($N = 27$) to state their preferred method in forced binary choices across different concerns: layout control, reconstruction faithfulness to the template, and their overall preferred scene. For each concern and method, we presented users with three scene renderings.

Win Rate (%)			
Layout Control	**Faithfulness of 3D Reconstruction to Template**		
SynCity [15]	TRELLIS [62]	TripoSG [35]	Hunyuan3D-2.1 [24]
100.0	71.6	100.0	100.0
Generated Scene Preference Win Rate (%)			
SynCity	NuiScene [31]	3DTown [80]	TRELLIS w/ our template
63.0	74.1	59.3	78.6

Table 3: Geometric reconstruction quality. We evaluate the geometric reconstruction quality of TRELLIS [62] and our method on synthetic scenes using the Chamfer Distance, F-score, LPIPS, and PSNR. We use the template size as described in our experimental details, as well as a larger one, with 25 scenes each.

Method	Chamfer ↓	F-score ↑	LPIPS ↓	PSNR ↑
Template Size: 1344×672 px				
TRELLIS	**0.0137**	0.6685	0.5628	12.2615
Ours	0.0166	**0.7029**	**0.5340**	**13.3197**
Template Size: 2240×1120 px				
TRELLIS	0.0392	0.6967	**0.5762**	11.6361
Ours	**0.0302**	**0.7536**	0.5812	**12.6063**

For general generation preference, we asked users to pick a favorite scene in a binary forced-choice setting. For fair comparison, we built scenes with prompts similar to those generated by NuiScene [31] and 3DTown [80]. For SynCity, we used the same prompt (and applied no layout constraints), see Fig. 6. Scenes produced by SynCity 3000 were overall preferred.

While we evaluated template faithfulness and geometric quality, judging multi-view visual *plausibility* is not as straightforward in the absence of ground-truth data. Hence, we asked users to rate generated scenes from very implausible (1) to very plausible (5) and obtained an average rating of 3.57.

Ablations. We further show the effectiveness of our design choices in Tab. 1. Central to our method is the fine-tuning of models presented in [62] to enable convolutional inference. By using the models off the shelf, performance suffers significantly. This can be mainly attributed to the fact that the sparse structure model tries to infer a canonical front-facing orientation, which is ill-defined for

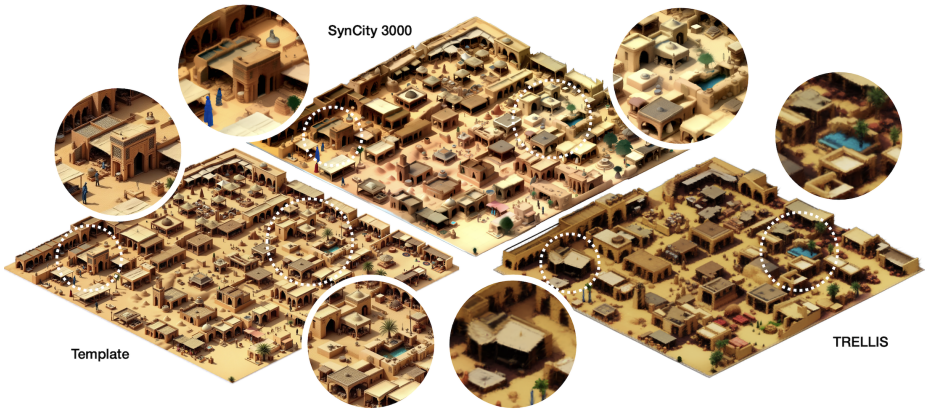

Fig. 7: Large Scene Detail Comparison. We let both our method (top) and TRELLIS (right) reconstruct a larger scene (3136 × 1568 pixels template, left). While the overall scene produced by TRELLIS looks decent, its resolution notably impacts its capability to reproduce details. Meanwhile, SynCity 3000 has a high adherence to the template and reproduces details more faithfully. We recommend viewing on a display and zooming in.

objects that lack this directionality. Without context, there is a minor deterioration in quality, indicating that the model benefits from information about the surrounding voxels. This is also true if we reduce the context ($V = 4/16$ for stage 1 and 2, respectively), indicating that the model captures long-range dependencies. If we use a large stride (here, an entire patch, which leads to no averaging of predictions), the reconstructions are also inferior. Smaller steps, and thus averaging tile predictions, help resolve ambiguities arising from template masking. For a single patch, structures in the background may be visible if they are not occluded by those in the foreground. This can lead to duplicated structures, whose occurrence is significantly reduced with smaller strides.

6 Conclusion

We have introduced SynCity 3000, a method to generate large-scale 3D scenes from scratch. Starting from a straightforward approach to describe arbitrarily complex scene layouts, we use a shifted-window approach to turn them into 2D templates, which are then converted into 3D Gaussian Splats through convolutional inference by a fine-tuned two-stage 3D generative model. We fine-tune this model with data generated by a synthetic dataset engine, which creates random scene-like 3D structures. By first generating the complete scene as a template in a joint diffusion process, we ensure semantic and visual coherence, directly addressing the limitations of SynCity. Furthermore, by fine-tuning the 3D generative model, we maintain its high-quality predictions while eliminating the need for brittle heuristics required by previous methods for scene generation.

Ethics. For details on ethics, data protection, and copyright, please see https://www.robots.ox.ac.uk/~vedaldi/research/union/ethics.html.

Acknowledgments. The authors of this work are supported by ERC 101001212-UNION and Meta Research. Special thanks to Robin Y. Park for her technical contributions.

References

1. AlimamaCreative: Flux-controlnet-inpainting. https://github.com/alimama-creative/FLUX-Controlnet-Inpainting (2024), gitHub repository
2. Ardelean, A., Özer, M., Egger, B.: Gen3dsr: Generalizable 3d scene reconstruction via divide and conquer from a single view. arXiv **2404.03421** (2024)
3. Armeni, I., He, Z.Y., Gwak, J., Zamir, A.R., Fischer, M., Malik, J., Savarese, S.: 3d scene graph: A structure for unified semantics, 3d space, and camera. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 5664–5673 (2019)
4. Bae, G., Budvytis, I., Cipolla, R.: Irondepth: Iterative refinement of single-view depth using surface normal and its uncertainty. In: British Machine Vision Conference (BMVC) (2022)
5. Bar-Tal, O., Yariv, L., Lipman, Y., Dekel, T.: Multidiffusion: Fusing diffusion paths for controlled image generation. arXiv **2302.08113** (2023)
6. Bar-Tal, O., Yariv, L., Lipman, Y., Dekel, T.: MultiDiffusion: Fusing diffusion paths for controlled image generation. arXiv.cs **abs/2302.08113** (2023)
7. Bhat, S.F., Birkl, R., Wofk, D., Wonka, P., Müller, M.: Zoedepth: Zero-shot transfer by combining relative and metric depth. arXiv **2302.12288** (2023)
8. Chen, D.Z., Li, H., Lee, H.Y., Tulyakov, S., Nießner, M.: Scenetex: High-quality texture synthesis for indoor scenes via diffusion priors. In: Proc. CVPR (2024)
9. Chen, Y., Viégas, F., Wattenberg, M.: Beyond surface statistics: Scene representations in a latent diffusion model. arXiv **2306.05720** (2023)
10. Chung, J., Lee, S., Nam, H., Lee, J., Lee, K.M.: LucidDreamer: Domain-free generation of 3d gaussian splatting scenes. In: arXiv (2023)
11. Collins, J., Goel, S., Deng, K., Luthra, A., Xu, L., Gundogdu, E., Zhang, X., Vicente, T.F.Y., Dideriksen, T., Arora, H., Guillaumin, M., Malik, J.: ABO: Dataset and benchmarks for real-world 3D object understanding. In: Proc. CVPR. pp. 21126–21136 (2022)
12. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In: Proc. CVPR (2017)
13. Dai, T., Wong, J., Jiang, Y., Wang, C., Gokmen, C., Zhang, R., Wu, J., Fei-Fei, L.: Automated creation of digital cousins for robust policy learning. arXiv **2410.07408** (2024)
14. Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S.Y., VanderBilt, E., Kembhavi, A., Vondrick, C., Gkioxari, G., Ehsani, K., Schmidt, L., Farhadi, A.: Objaverse-XL: A universe of 10M+ 3D objects. CoRR **abs/2307.05663** (2023)
15. Engstler, P., Shtedritski, A., Laina, I., Rupprecht, C., Vedaldi, A.: SynCity: Training-free generation of 3D worlds. In: Proceedings of the International Conference on Computer Vision (ICCV) (2025)

16. Engstler, P., Vedaldi, A., Laina, I., Rupprecht, C.: Invisible stitch: Generating smooth 3d scenes with depth inpainting. In: International Conference on 3D Vision (2025)
17. Feng, W., Zhu, W., Fu, T.j., Jampani, V., Akula, A., He, X., Basu, S., Wang, X.E., Wang, W.Y.: Layoutgpt: Compositional visual planning and generation with large language models. Advances in Neural Information Processing Systems **36**, 18225–18250 (2023)
18. Fu, H., Jia, R., Gao, L., Gong, M., Zhao, B., Maybank, S., Tao, D.: 3d-future: 3d furniture shape with texture. International Journal of Computer Vision **129**(12), 3313–3337 (2021)
19. Gao, D., Rozenberszki, D., Leutenegger, S., Dai, A.: DiffCAD: weakly-supervised probabilistic CAD model retrieval and alignment from an RGB image. arXiv **2311.18610** (2024)
20. Gümeli, C., Dai, A., Nießner, M.: ROCA: Robust CAD model retrieval and alignment from a single image. In: Proc. CVPR (2022)
21. Höllein, L., Cao, A., Owens, A., Johnson, J., Nießner, M.: Text2room: Extracting textured 3d meshes from 2d text-to-image models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7909–7920 (2023)
22. Hu, Z., Iscen, A., Jain, A., Kipf, T., Yue, Y., Ross, D.A., Schmid, C., Fathi, A.: SceneCraft: An LLM agent for synthesizing 3d scenes as Blender code. In: Proc. ICML (2024)
23. Huang, Z., Guo, Y.C., An, X., Yang, Y., Li, Y., Zou, Z.X., Liang, D., Liu, X., Cao, Y.P., Sheng, L.: Midi: Multi-instance diffusion for single image to 3D scene generation. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 23646–23657 (2025)
24. Hunyuan3D, T., Yang, S., Yang, M., Feng, Y., Huang, X., Zhang, S., He, Z., Luo, D., Liu, H., Zhao, Y., Lin, Q., Lai, Z., Yang, X., Shi, H., Zhao, Z., Zhang, B., Yan, H., Wang, L., Liu, S., Zhang, J., Chen, M., Dong, L., Jia, Y., Cai, Y., Yu, J., Tang, Y., Guo, D., Yu, J., Zhang, H., Ye, Z., He, P., Wu, R., Wei, S., Zhang, C., Tan, Y., Sun, Y., Niu, L., Huang, S., Zheng, B., Liu, S., Chen, S., Yuan, X., Yang, X., Liu, K., Zhu, J., Chen, P., Liu, T., Wang, D., Liu, Y., Linus, Jiang, J., Huang, J., Guo, C.: Hunyuan3D 2.1: From images to high-fidelity 3D assets with production-ready PBR material. arXiv **2506.15442** (2025)
25. International Commission on Illumination: Colorimetry — part 4: Cie 1976 l*a*b* colour space. Standard, International Commission on Illumination, Vienna, AT (1976)
26. Ke, B., Obukhov, A., Huang, S., Metzger, N., Daudt, R.C., Schindler, K.: Repurposing diffusion-based image generators for monocular depth estimation. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 9492–9502 (2024)
27. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
28. Khanna, M., Mao, Y., Jiang, H., Haresh, S., Shacklett, B., Batra, D., Clegg, A., Undersander, E., Chang, A.X., Savva, M.: Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16384–16393 (2024)
29. Kim, S.W., Brown, B., Yin, K., Kreis, K., Schwarz, K., Li, D., Rombach, R., Torralba, A., Fidler, S.: Neuralfield-ldm: Scene generation with hierarchical latent diffusion models. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2023)

30. Kuo, W., Angelova, A., Lin, T.Y., Dai, A.: Patch2CAD: Patchwise embedding learning for in-the-wild shape retrieval from a single image. In: Proc. ICCV (2021)
31. Lee, H.H., Han, Q., Chang, A.X.: NuiScene: exploring efficient generation of unbounded outdoor scenes. In: Proc. ICCV (2025)
32. Lee, J., Im, W., Lee, S., Yoon, S.E.: Diffusion probabilistic models for scene-scale 3d categorical data. arXiv **2301.00527** (2023)
33. Lee, J., Lee, S., Jo, C., Im, W., Seon, J., Yoon, S.E.: Semcity: Semantic scene generation with triplane diffusion. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 28337–28347 (2024)
34. Li, W., Cai, F., Mi, Y., Yang, Z., Zuo, W., Wang, X., Fan, X.: SceneDreamer360: text-driven 3D-consistent scene generation with panoramic Gaussian splatting. arXiv **2408.13711** (2024)
35. Li, Y., Zou, Z.X., Liu, Z., Wang, D., Liang, Y., Yu, Z., Liu, X., Guo, Y.C., Liang, D., Ouyang, W., Cao, Y.P.: TripoSG: high-fidelity 3D shape synthesis using large-scale rectified flow models. arXiv **2502.06608** (2025)
36. Li, Y., Zou, Z.X., Liu, Z., Wang, D., Liang, Y., Yu, Z., Liu, X., Guo, Y.C., Liang, D., Ouyang, W., et al.: Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. arXiv **2502.06608** (2025)
37. Li, Y., Liu, H., Wu, Q., Mu, F., Yang, J., Gao, J., Li, C., Lee, Y.J.: GLIGEN: open-set grounded text-to-image generation. In: Proc. CVPR (2023)
38. Li, Z., Wang, Y., Zheng, H., Luo, Y., Wen, B.: Sparc3d: Sparse representation and construction for high-resolution 3d shapes modeling. arXiv **2505.14521** (2025)
39. Lin, C., MU, Y.: InstructScene: instruction-driven 3d indoor scene synthesis with semantic graph prior. In: International Conference on Learning Representations (ICLR) (2024)
40. Lin, C., Mu, Y.: Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. arXiv **2402.04717** (2024)
41. Meng, Q., Li, L., Nießner, M., Dai, A.: Lt3sd: Latent trees for 3d scene diffusion. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 650–660 (2025)
42. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
43. Ocal, B.M., Tatarchenko, M., Karaoglu, S., Gevers, T.: SceneTeller: language-to-3D scene generation. In: Proc. ECCV (2024)
44. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv **2304.07193** (2023)
45. Ouyang, H., Heal, K., Lombardi, S., Sun, T.: Text2Immersion: Generative immersive scene with 3D gaussians. arXiv.cs **abs/2312.09242** (2023)
46. Pan, X., Charron, N., Yang, Y., Peters, S., Whelan, T., Kong, C., Parkhi, O., Newcombe, R., Ren, Y.C.: Aria digital twin: A new benchmark dataset for egocentric 3d machine perception. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 20133–20143 (2023)
47. Po, R., Wetzstein, G.: Compositional 3d scene generation using locally conditioned diffusion. ArXiv **abs/2303.12218** (2023), https://api.semanticscholar.org/CorpusID:257663283
48. Ramakrishnan, S.K., Gokaslan, A., Wijmans, E., Maksymets, O., Clegg, A., Turner, J., Undersander, E., Galuba, W., Westbury, A., Chang, A.X., et al.: Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. arXiv **2109.08238** (2021)

49. Ren, X., Huang, J., Zeng, X., Museth, K., Fidler, S., Williams, F.: Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 4209–4219 (2024)
50. Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M.A., Paczan, N., Webb, R., Susskind, J.M.: Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10912–10922 (2021)
51. Shriram, J., Trevithick, A., Liu, L., Ramamoorthi, R.: Realmdreamer: Text-driven 3d scene generation with inpainting and depth diffusion. arXiv **2404.07199** (2024)
52. Stan, G.B.M., Wofk, D., Fox, S., Redden, A., Saxton, W., Yu, J., Aflalo, E., Tseng, S.Y., Nonato, F., Muller, M., et al.: Ldm3d: Latent diffusion model for 3d. arXiv **2305.10853** (2023)
53. Sun, F.Y., Liu, W., Gu, S., Lim, D., Bhat, G., Tombari, F., Li, M., Haber, N., Wu, J.: LayoutVLM: differentiable optimization of 3d layout via vision-language models. arXiv **2412.02193** (2025)
54. Tang, J., Nie, Y., Markhasin, L., Dai, A., Thies, J., Nießner, M.: DiffuScene: denoising diffusion models for generative indoor scene synthesis. In: Proc. CVPR (2024)
55. Tang, J., Nie, Y., Markhasin, L., Dai, A., Thies, J., Nießner, M.: Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 20507–20518 (2024)
56. Team, T.H.: Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material (2025)
57. Wang, G., Wang, P., Chen, Z., Wang, W., Loy, C.C., Liu, Z.: Perf: Panoramic neural radiance field from a single panorama. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2024)
58. Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., Novotny, D.: VGGT: Visual geometry grounded transformer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2025)
59. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., Zhu, J.: ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. arXiv.cs **abs/2305.16213** (2023)
60. Wu, T., Zheng, C., Cham, T.J.: Panodiffusion: 360-degree panorama outpainting via diffusion. arXiv **2307.03177** (2023)
61. Wu, Z., Li, Y., Yan, H., Shang, T., Sun, W., Wang, S., Cui, R., Liu, W., Sato, H., Li, H., Ji, P.: BlockFusion: Expandable 3D scene generation using latent tri-plane extrapolation. arXiv.cs (2024)
62. Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., Yang, J.: Structured 3d latents for scalable and versatile 3d generation. arXiv **2412.01506** (2024)
63. Xie, D., Bi, S., Shu, Z., Zhang, K., Xu, Z., Zhou, Y., Pirk, S., Kaufman, A., Sun, X., Tan, H.: LRM-Zero: training large reconstruction models with synthesized data. In: Proc. NeurIPS (2024)
64. Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., Zhao, H.: Depth anything: Unleashing the power of large-scale unlabeled data. arXiv **2401.10891** (2024)
65. Yang, X., Man, Y., Chen, J.K., Wang, Y.X.: Scenecraft: Layout-guided 3d scene generation. In: Advances in Neural Information Processing Systems (2024)

66. Yang, Y., Sun, F.Y., Weihs, L., VanderBilt, E., Herrasti, A., Han, W., Wu, J., Haber, N., Krishna, R., Liu, L., Callison-Burch, C., Yatskar, M., Kembhavi, A., Clark, C.: Holodeck: Language guided generation of 3d embodied AI environments. In: Proc. CVPR (2024)
67. Yao, K., Zhang, L., Yan, X., Zeng, Y., Zhang, Q., Xu, L., Yang, W., Gu, J., Yu, J.: Cast: Component-aligned 3D scene reconstruction from an RGB image. ACM Transactions on Graphics (TOG) **44**(4), 1–19 (2025)
68. Yeshwanth, C., Liu, Y.C., Nießner, M., Dai, A.: Scannet++: A high-fidelity dataset of 3d indoor scenes. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12–22 (2023)
69. Yu, H.X., Duan, H., Herrmann, C., Freeman, W.T., Wu, J.: Wonderworld: Interactive 3d scene generation from a single image. arXiv **2406.09394** (2024)
70. Yu, H.X., Duan, H., Hur, J., Sargent, K., Rubinstein, M., Freeman, W.T., Cole, F., Sun, D., Snavely, N., Wu, J., et al.: Wonderjourney: Going from anywhere to everywhere. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6658–6667 (2024)
71. Zhai, G., Örnek, E.P., Chen, D.Z., Liao, R., Di, Y., Navab, N., Tombari, F., Busam, B.: EchoScene: indoor scene generation via information echo over scene graph diffusion. In: Proc. ECCV (2024)
72. Zhai, G., Örnek, E.P., Chen, D.Z., Liao, R., Di, Y., Navab, N., Tombari, F., Busam, B.: Echoscene: Indoor scene generation via information echo over scene graph diffusion. In: European Conference on Computer Vision. pp. 167–184. Springer (2024)
73. Zhan, G., Zheng, C., Xie, W., Zisserman, A.: A general protocol to probe large vision models for 3d physical understanding. In: The Thirty-eighth Annual Conference on Neural Information Processing Systems (2024)
74. Zhang, B., Tang, J., Nießner, M., Wonka, P.: 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. ACM Trans. Graph. **42**(4) (jul 2023). https://doi.org/10.1145/3592442, https://doi.org/10.1145/3592442
75. Zhang, J., Li, X., Wan, Z., Wang, C., Liao, J.: Text2NeRF: Text-driven 3D scene generation with neural radiance fields. arXiv.cs **abs/2305.11588** (2023)
76. Zhang, L., Wang, Z., Zhang, Q., Qiu, Q., Pang, A., Jiang, H., Yang, W., Xu, L., Yu, J.: CLAY: A controllable large-scale generative model for creating high-quality 3D assets. In: Proc. SIGGRAPH (2024)
77. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proc. CVPR (2018)
78. Zhao, Z., Lai, Z., Lin, Q., Zhao, Y., Liu, H., Yang, S., Feng, Y., Yang, M., Zhang, S., Yang, X., Shi, H., Liu, S., Wu, J., Lian, Y., Yang, F., Tang, R., He, Z., Wang, X., Liu, J., Zuo, X., Chen, Z., Lei, B., Weng, H., Xu, J., Zhu, Y., Liu, X., Xu, L., Hu, C., Huang, T., Wang, L., Zhang, J., Chen, M., Dong, L., Jia, Y., Cai, Y., Yu, J., Tang, Y., Zhang, H., Ye, Z., He, P., Wu, R., Zhang, C., Tan, Y., Xiao, J., Tao, Y., Zhu, J., Xue, J., Liu, K., Zhao, C., Wu, X., Hu, Z., Qin, L., Peng, J., Li, Z., Chen, M., Zhang, X., Niu, L., Wang, P., Wang, Y., Kuang, H., Fan, Z., Zheng, X., Zhuang, W., He, Y., Liu, T., Yang, Y., Wang, D., Liu, Y., Jiang, J., Huang, J., Guo, C.: Hunyuan3D 2.0: Scaling diffusion models for high resolution textured 3d assets generation. arXiv **2501.12202** (2025)
79. Zhao, Z., Lai, Z., Lin, Q., Zhao, Y., Liu, H., Yang, S., Feng, Y., Yang, M., Zhang, S., Yang, X., et al.: Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. arXiv **2501.12202** (2025)
80. Zheng, K., Zhang, R., Gu, J., Yang, J., Wang, X.E.: Constructing a 3D town from a single image. arXiv **2505.15765** (2025)

81. Zheng, K., Zhang, R., Gu, J., Yang, J., Wang, X.E.: Constructing a 3d town from a single image. arXiv **2505.15765** (2025)
82. Zhou, X., Ran, X., Xiong, Y., He, J., Lin, Z., Wang, Y., Sun, D., Yang, M.H.: Gala3d: Towards text-to-3d complex scene generation via layout-guided generative gaussian splatting. arXiv **2402.07207** (2024)