ExCellGen: Fast, Controllable, Photorealistic 3D Scene Generation from a Single Real-World Exemplar

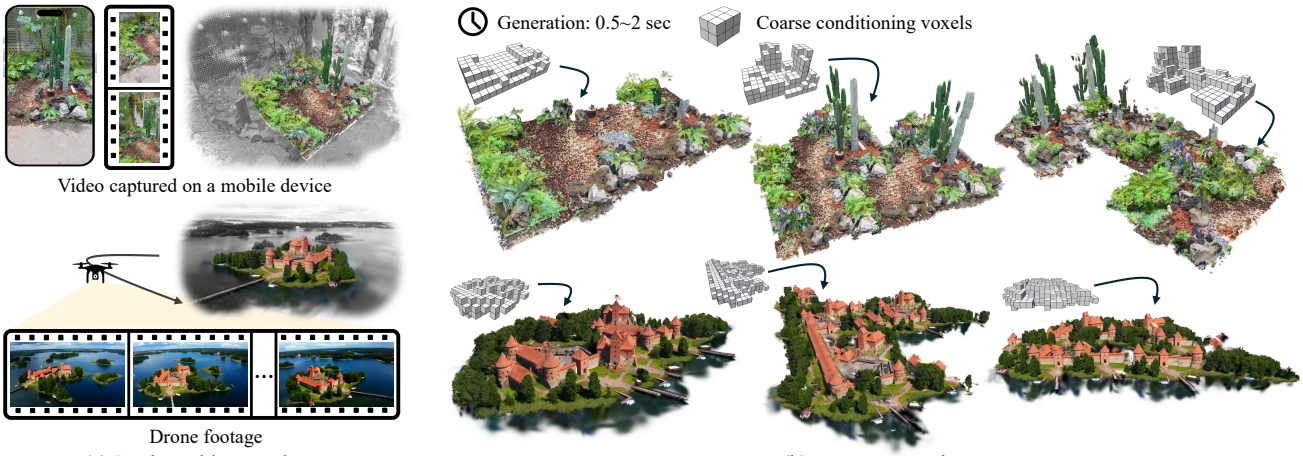
Clément Jambon*
Massachusetts Institute of Technology
Cambridge, USA
cjambon@mit.edu

Changwoon Choi
Seoul National University
Seoul, South Korea
changwoon.choi00@gmail.com

Dongsu Zhang
Seoul National University
Seoul, South Korea
96lives@gmail.com

Olga Sorkine-Hornung
ETH Zurich
Zurich, Switzerland
sorkine@inf.ethz.ch

Young Min Kim
Seoul National University
Seoul, South Korea
youngmin.kim@snu.ac.kr



(a) Real-world exemplar (b) Scene generation

Figure 1. We propose ExCellGen, a framework for fast, controllable, and photorealistic 3D scene generation from real-world exemplars. We convert casual video inputs (e.g., drone or mobile footage) into a high-quality 3D representation using 3D Gaussian splats. Within our GUI editor, users can extract regions from which they wish to generate variations. Our two-stage amortized generation strategy first uses a per-scene Generative Cellular Automaton (GCA) [75] to produce a sparse volume of featurized voxels, then composites the final appearance via sparse patch-based remapping from the original scene. Given coarse conditioning voxels, users can synthesize new variations in just seconds in a fully interactive way.

Abstract

Photorealistic 3D scene generation is challenging due to the scarcity of large-scale, high-quality real-world 3D datasets and complex workflows requiring specialized expertise for manual modeling. These constraints often result in slow iteration cycles, where each modification demands substantial effort, ultimately stifling creativity. We propose a fast, exemplar-driven framework for generating 3D scenes from a single casual input, such as handheld video or drone footage. Our method first leverages 3D Gaussian Splatting (3DGS) to robustly reconstruct input scenes with a high-quality 3D appearance model. We then train a per-scene Generative Cellular Automaton (GCA) to produce a sparse volume of featurized voxels, effectively amortizing scene generation while enabling controllability. A subsequent patch-based remapping step composites the complete scene from the exemplar's initial 3D Gaussian splats, successfully recovering the appearance statistics of the input scene. The entire pipeline can be trained in less than 10 minutes for each exemplar and generates scenes in 0.5-2 seconds. Our method enables interactive creation with full user control, and we showcase complex 3D generation results from real-world exemplars within a self-

*Work done at Seoul National University and ETH Zurich.

contained interactive GUI.

Keywords: ***Interactive Method, Generative Cellular Automata, 3D Gaussian Splatting, Scene Generation, Patch-based Synthesis***

1. Introduction

Generating realistic 3D scenes is key to many applications in 3D content creation, visual effects, robotics [72], and autonomous driving [43]. Generative modeling has recently demonstrated remarkable progress in creating photorealistic images, largely owing to the tremendous data and development of deep learning network architectures. In contrast, achieving realism in 3D scene generation faces several fundamental challenges. Unlike images or photos, 3D models are limited in scale and quality. Available 3D datasets are often synthetic, and composed of object-centric meshes with simple textured appearance [13]. In contrast, existing real-world datasets are mostly confined to specific distributions, including indoor scenes [8], urban footage [33] or object-centric captures [52, 36, 17]. This falls short of the diversity found in real-world scenes, rich in intricate geometry and natural elements like vegetation, which are notoriously difficult to capture with meshes or represent using conventional 3D assets.

In this work, we propose to formulate the task of 3D scene generation as an exemplar-based synthesis task, building on a rich line of research in texture synthesis [15, 19]. We target real-world captures in the form of casual video footage. All of our input exemplars are derived from footage recorded using smartphones or drones, demonstrating the practicality of our approach. To reconstruct 3D scenes, we build on the recent success of 3D Gaussian Splatting (3DGS) [28]. Gaussian splats enable fast retrieval and rendering and lead to a photorealistic appearance. They accurately reflect the user's intent by faithfully reproducing the images captured by the user, providing unambiguous control in selecting the ingredients of scene generation. Moreover, 3D Gaussians are explicit representations that can be directly manipulated, transported, and composited.

Previous approaches for exemplar-based synthesis demonstrate results on clean synthetic or pre-processed scenes, and typically support only mesh-like or SDF-based exemplars [38, 67, 66]. Even when supporting radiance fields, methods such as Sin3DGen [32] still require a mesh to guide generation. This requirement limits applicability to real-world scenes, where appearance may be fuzzy in some regions. Moreover, casually captured scenes often contain complex backgrounds and surrounding objects with unclear segmentation boundaries. We thus design intuitive means to select a desired region from the input exemplar scene by distilling and clustering semantic feature maps from large vision models [7]. This provides both control and additional guidance during generation.

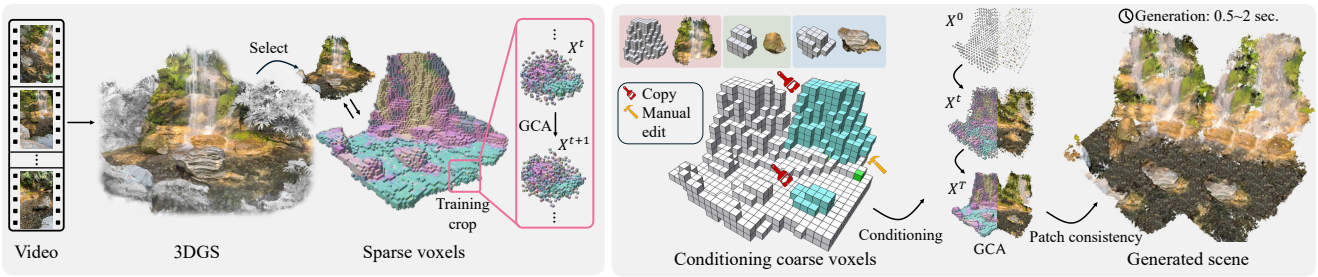
Synthesizing 3D scenes is often memory and computationally intensive, as it requires inferring 3D geometry, leading to slow, iterative processes (e.g., 1˜3 minutes for Sin3DGen). We propose a fast and controllable two-stage generation strategy that enables interactive feedback within seconds. First, we generate a sparse voxel volume that captures the mesoscale structure of the scene, abstracting away fine-grained details and amortizing the generation task. Then, we reconstruct high-frequency appearance by compositing 3D Gaussians from the exemplar, guided by the generated voxels.

More precisely, in the first stage, we adapt Generative Cellular Automata [75] (GCA) to controllable scene generation to efficiently generate a sparse 3D volume of voxels. The generative kernels of GCA are trained from a single exemplar and are highly efficient in both training (less than 10 minutes) and generation (0.5s-2s). This neural approach offers two additional advantages. First, contrary to cascaded generation strategies [32], our generative network is naturally trained to reflect conditional signals, which we materialize as coarse voxels that describe the location and shape of the desired output scene. Second, neural approaches generalize better across various distributions without the need for cumbersome parameter tuning. In the second stage, we recover the final appearance of the scene by introducing a novel sparse and efficient patch-based consistency step, which results in smooth transitions between voxels. The final scene is converted into 3DGS with high-frequency details by transporting and compositing 3D Gaussians from the exemplar scene according to the generated voxels. We demonstrate the practicality and capabilities of our method on a variety of real-world scenes captured using smartphones or drones, and manipulated directly within our interactive GUI editor.

In summary, our contributions are as follows: (a) We propose an efficient 3D scene generation technique in which exemplar scenes are directly captured from casual real-world videos; (b) Our method is controllable, inviting user controls in capturing, selecting, and compositing the scene; (c) We formulate a lightweight generative model from a single exemplar, allowing scene synthesis at interactive rates (approximately 0.5-2s); (d) We provide a fully self-contained GUI to demonstrate the practicality of the method. The source code is available at https://github.com/clementjambon/excellgen.

2. Related Works

3D scene representations Generating high-quality 3D scenes has long been a central challenge in computer graphics, hinging critically on the choice of 3D representation. Unlike images, 3D scenes must capture not only appearance but also geometry and structural details. Discrete rep-



(a) Training from a single exemplar

(b) Scene generation

Figure 2. Overview of our pipeline. *Training phase.* (left) From a casual video, an exemplar scene is reconstructed as 3D Gaussians augmented with DINO features. After selecting the region of interest within an interactive editor, the 3D Gaussians are converted into a sparse volume of voxels. A per-scene Generative Cellular Automaton (GCA) is then efficiently trained on random crops of the scenes in under 10 minutes. *Generation phase.* (right) A set of coarse conditioning voxels, either copied from parts of the exemplar or manually edited, is provided to the pre-trained GCA. The GCA generates a novel sparse volume of featurized voxels. These voxels are then remapped to the exemplar's 3D Gaussians using a sparse patch-based consistency step.

resentations such as meshes and point clouds offer explicit control, enabling direct manipulation and composition [78]. However, their irregular and heterogeneous nature complicates generation. To address this, continuous field-based representations like signed distance functions (SDFs) [47] and occupancy fields [39, 10], with an optional color field for modeling appearance, have been widely adopted for scene generation. Yet, by enforcing the constraint of a surface, they struggle to model fine details (e.g., fuzzy surfaces, foliage, etc). Neural Radiance Fields (NeRFs) have recently revolutionized 3D scene reconstruction by modeling scenes as continuous volumetric representations [40, 4]. Notably, they can be trained within minutes from casual, real-world video footage [41, 21]. However, their dense nature poses challenges for editing and recomposition, often requiring auxiliary structures (e.g., cages [26, 74]) or post-processing to merge parts seamlessly [24]. 3D Gaussian splatting (3DGS) [28] alleviates these limitations by representing scenes as collections of fuzzy Gaussians, achieving impressive rendering speed, quality, and robustness to casual inputs. Crucially, Gaussians blend naturally, enabling smooth transitions and composability. Given these advantages, we adopt 3D Gaussians as our base representation. To efficiently generate and manipulate them, we employ sparse 3D voxel grids, which abstract away their irregular spatial distribution. Sparse voxel grids have long been used as a scalable 3D representation [12] and are increasingly prominent in generative modeling [53, 54, 69, 38].

Controllable 3D scene generation Recent advances in 2D generative models [55] and the emergence of large-scale 3D datasets [13] have significantly accelerated progress in 3D scene generation. A line of works leverages 2D diffusion priors via Score Distillation Sampling [50, 34], enabling text-driven generation with controllability through bounding boxes [48] or automatically learned layouts [20]. Additional guidance can come from coarse shape inputs [16], sketches [35, 9], or exemplar-based fine-tuning [65]. While there have been notable gains in efficiency [71], fidelity [37], and the use of multiview priors [59], these methods remain computationally intensive and largely limited to simple, object-centric scenes. In contrast, recent feedforward approaches conditioned on input images [69, 60] offer faster inference, yet often rely on spherical viewpoint assumptions [11, 73], produce unrealistic object-level assets [69], or lack true geometric consistency [61]. Alternative methods condition generation on scene layouts [68, 5], providing greater structural control. However, they typically require domain-specific datasets and extensive supervision, limiting their applicability to constrained environments such as urban landscapes [14, 70] or indoor scenes [57, 1].

Exemplar-based methods Exemplar-based methods have a rich history in Computer Graphics, with applications including texture synthesis [19, 18], image analogies [25], and part-based shape modeling [22]. Many of these approaches are grounded in patch-based synthesis or optimization techniques [27, 3]. Patch-based synthesis algorithms alternate repetitively between two stages: (a) an exact or approximate matching algorithm pairs a patch of the target representation to a patch in the exemplar, (b) patches from the exemplar are aggregated to produce a new target representation. A key contribution in this space is the *PatchMatch* algorithm [2], which introduced a fast, randomized method to amortize the patch-matching process. Sin3DGen [32] adapted this strategy to 3D scene generation, using a hierarchical coarse-to-fine approach to synthesize radiance volumes. However, because it relies on patch-based statistics in a purely top-down manner [64], it struggles to capture high-level semantics and structural coherence. Furthermore, it operates densely over the full

radiance field, resulting in long generation times (1˜3 minutes). Recently, 3D Gaussian Splat brushes [45] were introduced, enabling interactive brush-based painting from real-world scenes. However, the method remains limited to curve- or stroke-based control, constraining both interaction and generated outputs to a 2.5D appearance.

Another line of work leverages recent deep learning advances by training neural networks on a single exemplar, effectively amortizing inference time through an upfront training phase. These methods, both in 2D [42, 58, 63] and in 3D [66, 67], use random augmentations and limited receptive fields to learn generative models capable of producing variations of the input exemplar. However, with few exceptions [38], these approaches often require long training times (on the order of hours) and struggle to generate realistic appearance. This limitation arises from explicitly modeling the output as point clouds [38], signed distance fields (SDFs) [66], or occupancy grids [67], which largely confines them to clean, mesh-like geometry and makes it challenging to handle the fuzzy regions often found in real-world scenes.

3. Method

We present a method for synthesizing realistic 3D scenes from a single in-the-wild exemplar. Figure 2 illustrates an overview of our pipeline. Starting from an input video, we reconstruct a 3D Gaussian splatting scene enriched with semantic features (Section 3.1). To obtain a more compact and generation-friendly representation, we abstract the irregular splats into a sparse voxel grid, where each cell encodes a feature vector that combines both appearance and semantic information (Section 3.2). The grid is synthesized in 0.5–2 seconds using Generative Cellular Automata (GCA), a lightweight generative model conditioned on coarse voxel inputs and trained per scene in less than 10 minutes (Section 3.3). Finally, we recover a photorealistic 3D scene by remapping Gaussians through a sparse, patch-based consistency step (Section 3.4).

3.1. Semantically Augmented 3D Gaussian Splatting

3D Gaussian splatting [28] represents scenes as a set of anisotropic 3D Gaussians with positions $\boldsymbol{\mu} \in \mathbb{R}^3$ and covariance matrices $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ parameterized by a scaling vector $\mathbf{s} \in \mathbb{R}^3$ and a rotation quaternion $\mathbf{r} \in \mathbb{R}^4$. Their appearance is described by an opacity coefficient $\eta \in \mathbb{R}$ and a view-dependent color $\mathbf{c} : \mathbb{S}^2 \to \mathbb{R}^3$ originally parameterized by spherical harmonics with coefficients $\boldsymbol{\gamma} \in \mathbb{R}^m$ (where m is typically 16 for spherical harmonics of degree 3). 3D Gaussians are rendered and optimized through a differentiable alpha-compositing rasterization algorithm as follows:

$$C = \sum_i \mathbf{c}^{(i)} \alpha^{(i)} \prod_{j=1}^{i-1} (1 - \alpha^{(j)}), \text{ where } \alpha^{(i)} = \eta^{(i)} G_{\text{2D}}^{(i)}(\mathbf{x}) \tag{1}$$

and $G_{\text{2D}}^{(i)}(\mathbf{x})$ is the 2D linearized Gaussian density kernel of 3D Gaussian i after projecting it on the 2D viewplane. Please refer to [28] for more details on the rendering algorithm.

To incorporate higher-level scene context and guide generation beyond raw appearance, we draw inspiration from previous works [62, 77, 30] and augment the unstructured mixture of independent Gaussians with semantic features $\mathbf{f} \in \mathbb{R}^d$. These features also enable interactive selection of subregions of the scene in our GUI (see Figure 5 and the supplemental video). In practice, we adopt DINO features [7, 44]. Initially extracted as 2D feature maps with 768 channels, we reduce their dimensionality to 8 via PCA, and lift them to 3D by replacing $\mathbf{c}^{(i)}$ with $\mathbf{f}^{(i)}$ in Equation (1). Additional implementation details are provided in the supplemental material (Section S.2.1).

3.2. Sparse Voxel Grids

In order to facilitate and accelerate scene generation, we abstract the set of 3D Gaussians in a sparse voxel grid $\mathbf{V} = \{\mathbf{p}_i\}_{i=1}^N$ where N is the number of voxels and $\mathbf{p}_i \in \mathbb{Z}^3$ is the positional index of an occupied voxel in 3D. Each voxel is enriched with a feature vector $\mathbf{z}_i$, forming a sparse feature volume $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N$ that captures both the semantic and appearance attributes of the 3D Gaussians it encompasses. This process is illustrated in Figure 2 (a).

More precisely, we instantiate voxels where 3D Gaussians exceed a predefined opacity threshold $\tau = 0.1$. We build the volume of voxels at a fine resolution $r_t = 64^3$, and we derive a coarse volume at resolution $r_c = 16^3$ by downsampling the fine volume. The coarse volume is used as a conditioning signal (see Section 3.3), while we synthesize the scene at the more expressive fine resolution. The supplemental material (Section S.3.1) discusses the trade-offs between resolution choices. We maintain diversity among cells by selecting the features of the Gaussian with the highest opacity rather than simply averaging features from all Gaussians. We extract both the appearance encoded in the coefficients of spherical harmonics of the chosen splat and the semantic information of the distilled DINO features presented in Section 3.1. In practice, we use four dimensions for each of them, resulting in an 8-dimensional feature per voxel. For consistency, each of them is independently PCA-ed and subsequently re-normalized. Additional details are provided in the supplemental material (Section S.2).

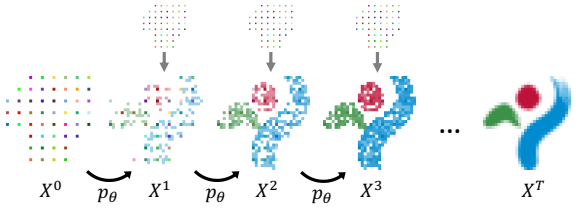


Figure 3. At each time step t, GCA samples a new state X^{t+1} composed of sparse voxel occupancies equipped with features from $p_\theta(X^{t+1} \mid X^t, X^0)$. Recursively applying the transition kernel p_θ to the initial state X^0 yields the final generated state X^T. We adapt GCA to conditional generation by choosing X^0 to be the upsampled coarse set of conditioning voxels and concatenating it to X^t each time p_θ is applied (gray arrows on the top).

3.3. Generating Sparse Voxels with GCA

To generate sparse voxels grids, we adapt *Generative Cellular Automata* (GCA) [75, 76]. At a high level, GCA only predicts occupied surface voxels and their corresponding features as described in the previous section.

3.3.1 Background: *Generative Cellular Automata*

GCA [75, 76] represents shapes as a sparse grid of voxels $\mathbf{V}$ equipped with per-voxel features $\mathbf{Z}$, effectively modeling the joint distribution $X = (\mathbf{V}, \mathbf{Z})$. As illustrated in Figure 3, starting from an initial state $X^0 = (\mathbf{V}^0, \mathbf{Z}^0)$, GCA generates a complete shape by iteratively sampling intermediate states $X^t = (\mathbf{V}^t, \mathbf{Z}^t)$ until reaching a final state X^T:

$$X^{t+1} \sim p_\theta(\cdot \mid X^t),$$

where T is the number of steps and p_θ is a learnable transition kernel. This kernel is implemented as a U-Net [56] with sparse convolutions, and only updates cells in the neighborhood of already occupied cells via local update rules similar to cellular automata. This inductive bias exploits the inherent connectivity and sparsity of 3D shapes, drastically reducing the search space in a high-resolution grid. For tractability, the kernel factorizes the joint probability of a state into separate terms for occupancy and features:

$$p_\theta(X^{t+1} \mid X^t) = \underbrace{p_\theta(\mathbf{V}^{t+1} \mid X^t)}_{(a)} \underbrace{p_\theta(\mathbf{Z}^{t+1} \mid X^t, \mathbf{V}^{t+1})}_{(b)} \tag{2}$$

where (a) is modeled as a Bernoulli distribution and (b) as a Gaussian distribution whose uncertainty decreases over time. GCA is trained through a process called *Infusion* [6]. The supplemental material (Section S.1) provides for a full description of GCA and a discussion of its training strategy.

3.3.2 Single-exemplar GCA and Controllable Generation

We adapt the generative framework of GCA to learn diverse yet plausible variations from a single input exemplar. To do so, we use a shallow network, greatly reducing the receptive field, drawing inspiration from prior work on single-image generative models [42, 63]. In addition, following prior work [66], we introduce random augmentations to enable the model to capture local structure more flexibly. Concretely, we use random voxel crops at fine resolution, each with side lengths between 25 and 30 and containing at least 250 voxels in total (via rejection sampling). To prevent GCA from learning the boundaries of these crops, we pad each crop by 2 voxels along every dimension; these padded voxels are not used to supervise the transition kernel but are provided as contextual input to the network. Implementation details are provided in the supplemental material (Section S.2.2) and ablation studies are discussed in Section 4.3.

Another key objective is to enable control over scene generation. Starting from the volume of voxels described in Section 3.2, we use its downsampled coarse version r_c both as the initial state X^0 and as a conditioning signal at every step (see Figure 3). During training, the conditioning volume r_c is derived from the ground-truth scene (i.e., via teacher forcing), while at inference time, it can be replaced with any valid voxel input. Note that generation occurs at the higher and more expressive resolution r_t, starting from features randomly initialized with a standard normal distribution. As discussed and ablated in the supplemental material (Section S.2.1), the choice of resolutions for r_c and r_t is critical for balancing controllability and output diversity.

3.4. Patch-based Consistent Scene Composition

A final patch-based consistency step converts the generated low-dimensional voxels into a scene with a rich photorealistic appearance composed of 3D Gaussian splats extracted from the input exemplar. If we naïvely copy and paste Gaussians in the cell with the closest feature vector from the exemplar scene, individual voxels are agnostic of their neighborhood, resulting in poor spatial consistency. We denote this approach as "Voxel-wise NN", and Figure 4 contains an example. We can enforce more consistency by considering a larger patch-wise context when selecting which cell to borrow the Gaussians from, denoted as "Patch-wise NN". However, this method breaks down when the voxels synthesized by GCA deviate too much from the input exemplar. To address this issue, we draw inspiration from patch-based image synthesis methods [3], introducing an efficient *sparse* patch-based consistency operation that redistributes the patch-wise statistics of the exemplar into a coherent spatial arrangement.

Patch-based synthesis was originally developed for dense 2D image domains, where simple per-pixel metrics

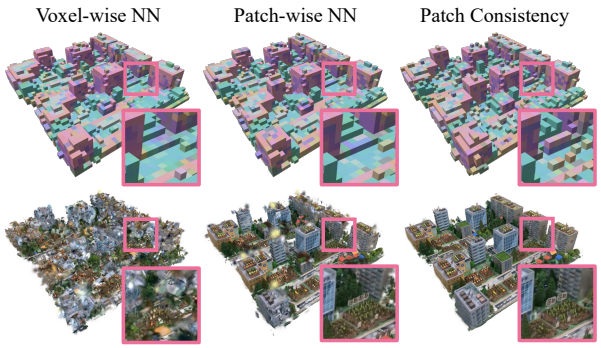


Figure 4. *Voxel-wise NN*. Naively remapping each predicted voxel to its nearest feature in the exemplar and filling in the corresponding 3D Gaussians fails to produce consistent results due to local consistency and the approximate predictions of GCA. *Patch-wise NN*. Using the patch-wise distance defined in Equation (3) improves visual coherence but still fails to account for modeling errors. *Patch Consistency*. We thus propose an additional sparse patch-based consistency operation to refine missing local statistics.

can be directly applied. In contrast, extending this idea to 3D requires accounting for voxel occupancy to track sparse geometry. We adapt it by defining a distance directly over sparse voxels, combined with a *voting mechanism* to locally grow or ungrow geometry moderately.

Notations and patch extraction. We refine the GCA output $X^0 = (\mathbf{V}^0, \mathbf{Z}^0)$ into a final state $X^K = (\mathbf{V}^K, \mathbf{Z}^K)$ through an iterative patch-based synthesis procedure, before reconstructing a photorealistic 3D Gaussian scene. Let $\mathbf{V}^k \subset \mathbb{Z}^3$ denote the occupied voxel coordinates at iteration k of this process, and $\mathbf{Z}^k = \{\mathbf{z}^k_\mathbf{p}\}_{\mathbf{p}\in\mathbf{V}^k}$ the corresponding voxel features. We define dense indicator and feature volumes:

$$O^k(\mathbf{p}) = \mathbb{I}[\mathbf{p} \in \mathbf{V}^k], \qquad F^k(\mathbf{p}) = \begin{cases} \mathbf{z}^k_\mathbf{p}, & \mathbf{p} \in \mathbf{V}^k, \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

Note that we use k instead of t to distinguish these iterative steps from the sampling steps of GCA.

Let $\mathcal{U} = \{-\frac{l-1}{2}, \dots, \frac{l-1}{2}\}^3$ be the set of patch offsets with $p = l^3$. For any voxel center $\mathbf{p}$ and state X^k, we define the associated patch

$$P_{X^k}(\mathbf{p}) = \big(O^k_\mathbf{p}, F^k_\mathbf{p}\big),$$
$$O^k_\mathbf{p}[\mathbf{u}] = O^k(\mathbf{p}+\mathbf{u}), \qquad F^k_\mathbf{p}[\mathbf{u}] = F^k(\mathbf{p}+\mathbf{u}).$$

for all $\mathbf{u} \in \mathcal{U}$. We denote by X^E the exemplar voxels.

Patch distance and matching. Given two patches P_e (resp. P_g) in the exemplar (resp. in the generated scene), we measure their similarity by

$$d(P_e, P_g) = (1-w)\, d_{\text{occ}}(P_e, P_g) + w\, d_{\text{feat}}(P_e, P_g), \quad (3)$$

where

$$d_{\text{occ}} = 1 - \frac{1}{p}\sum_{\mathbf{u}\in\mathcal{U}} O_e[\mathbf{u}]\, O_g[\mathbf{u}],$$

$$d_{\text{feat}} = 1 - \frac{\sum_{\mathbf{u}\in\mathcal{U}} O_e[\mathbf{u}]\, O_g[\mathbf{u}] \langle F_e[\mathbf{u}], F_g[\mathbf{u}]\rangle}{\sum_{\mathbf{u}\in\mathcal{U}} O_e[\mathbf{u}]\, O_g[\mathbf{u}]}.$$

At iteration k, each occupied voxel center $\mathbf{p} \in \mathbf{V}^k$ selects its best-matching exemplar patch center

$$\phi^k(\mathbf{p}) = \arg\min_{\mathbf{q}\in\mathbf{V}^E} d(P_{X^E}(\mathbf{q}), P_{X^k}(\mathbf{p}))\,. \quad (4)$$

Patch aggregation and voting. Each assignment $\phi^k(\mathbf{p})$ proposes a new exemplar patch. For any voxel $\mathbf{r}$ covered by the patch centered at $\mathbf{p}$, we define

$$\tilde{O}^k_\mathbf{p}(\mathbf{r}) = O^E\big(\phi^k(\mathbf{p}) + (\mathbf{r}-\mathbf{p})\big),$$
$$\tilde{F}^k_\mathbf{p}(\mathbf{r}) = F^E\big(\phi^k(\mathbf{p}) + (\mathbf{r}-\mathbf{p})\big).$$

Let $\mathcal{P}(\mathbf{r}) = \{\mathbf{p} \in \mathbf{V}^k : \mathbf{r} \in \mathbf{p} + \mathcal{U}\}$ be the set of patch centers whose patches overlap $\mathbf{r}$. We update voxel features by occupancy-weighted averaging,

$$F^{k+1}(\mathbf{r}) = \frac{\sum_{\mathbf{p}\in\mathcal{P}(\mathbf{r})} \tilde{O}^k_\mathbf{p}(\mathbf{r})\, \tilde{F}^k_\mathbf{p}(\mathbf{r})}{\sum_{\mathbf{p}\in\mathcal{P}(\mathbf{r})} \tilde{O}^k_\mathbf{p}(\mathbf{r})},$$

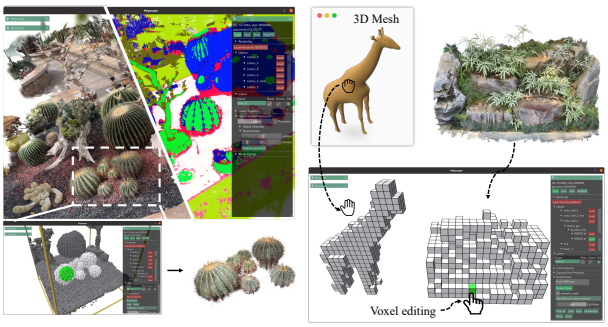
and compute an occupancy vote

$$v^k(\mathbf{r}) = \frac{1}{|\mathcal{P}(\mathbf{r})|}\sum_{\mathbf{p}\in\mathcal{P}(\mathbf{r})} \tilde{O}^k_\mathbf{p}(\mathbf{r}).$$

We keep voxels whose occupancy vote exceeds β (set to 0.5 in practice). We repeat this procedure (i.e., matching, followed by aggregation and voting) for K iterations.

Reconstruction of a 3D Gaussian scene. After K iterations, each synthesized occupied voxel $\mathbf{p} \in \mathbf{V}^K$ selects a single voxel from the exemplar using the distance in Equation (3) via the mapping in Equation (4). All Gaussians belonging to the selected exemplar voxel are then copied to the voxel corresponding to $\mathbf{p}$, without any further processing.

Discussion As shown in Figure 4, this procedure is key to recover missing geometric details. To allow only small refinements from the initial generated voxels, we limit additional voxels from being added farther than a distance λ_{patch}. We use a patch size of $l = 5$, 7 iterations and $\lambda_{\text{patch}} = 2$. Note that the distance in Equation (3) may assign high values to patches that are similar yet have low occupancy. In practice, we observed only minor adverse effects, as patches



(a) Interactive viewer and selection tool (b) Conditioning

Figure 5. Illustration of our user interface. (a) Our interactive viewer runs at real-time framerates (30-60fps) and comes with a selection tool using the quantized DINO features. Additional adjustments can be made at any stage with a manual selection tool. (b) Conditioning can be performed using parts of the exemplar, voxelized 3D meshes, or by manually editing voxels.

are updated for a limited number of iterations and the extent of geometric changes is restricted through λ_{patch}. The supplemental material (Section S.3.2) presents a discussion and ablation studies of these parameter choices. Section S.4 of the supplemental material highlights the differences between our approach and the related method Sin3DGen [32].

4. Results

We demonstrate our method's ability to generate high-quality and controllable 3D scenes from casually captured in-the-wild videos. First, we introduce our interactive user interface (Section 4.1), which allows users to import videos, select regions of interest, and author novel 3D scenes through intuitive composition of multiple exemplars. We then demonstrate the visual quality and controllability of generated scenes in diverse real-world environments (Section 4.2). Next, we ablate key components of our method (Section 4.3). Finally, we analyze the speed, showcasing its responsiveness and efficiency for interactive use (Section 4.4). Full implementation details are available in the supplemental material (Section S.2).

4.1. Interactive User Interface

We provide an interactive user interface that enables users to import videos, select regions of interest, and generate novel 3D scenes, as shown in Figure 5. Real-world scenes often contain cluttered backgrounds and lack clear object boundaries, motivating the need for an intuitive selection mechanism. To address this, we apply k-means clustering to the semantic features of 3D Gaussians, allowing users to interactively adjust the number of clusters and refine the selection through annotation or direct manipulation of 3D Gaussians (Figure 5(a), Figure S.2 in supplemental material and supplemental video).

We use a sparse voxel grid to provide the conditioning signal for generation, as illustrated in Figure 5 (b). In practice, the coarse geometry can be obtained from 1) coarse voxels from the existing exemplar, 2) input meshes, and/or 3) direct voxel editing. After generating each asset individually using these coarse voxels, our UI facilitates compositing them into a complete scene (see the supplemental video).

4.2. Scene Generation

We demonstrate that our method can generate diverse, controllable, and high-quality 3D scenes from casually captured real-world videos. All input videos are sourced from the LeRF dataset [29], YouTube, or recorded by ourselves, ensuring they reflect unconstrained real-world environments rather than synthetic or curated scenes. Figures 6 and 10 showcase examples generated from a single exemplar video conditioned on coarse voxel inputs. Given the same conditioning input, our method produces multiple plausible and visually distinct scenes within seconds, demonstrating its capacity for interactive and diverse content generation.

Beyond diversity, our method offers explicit controllability. Users can guide the synthesis process using coarse voxel geometry derived from exemplars, meshes, or manual edits, and can further manipulate results by transferring appearance to novel geometries, either from a mesh or another exemplar scene (Figure 7). This flexibility enables users to reconfigure and remix scene elements with minimal effort (see the supplemental video).

4.3. Ablations

A distinctive design choice of our method is that it operates entirely sparsely: we first grow a sparse set of voxels using GCA (Section 3.3), followed by local refinements via a sparse patch-based consistency step (Section 3.4). Figure 8 shows the impact of removing GCA from this pipeline. More precisely, we generate voxels hierarchically using our sparse patch-based optimization strategy. The coarsest resolution ($r_c = 16^3$) is initialized with the conditioning voxels (column 1) and random features. At each level, we apply the same parameters as described in Section 3.4, and upsample the volume by a factor of 2. This approach leads to repetitive patterns, lacks spatial and semantic consistency, and tends to inflate geometry beyond the conditioning input.

A crucial design choice to train GCA from a single exemplar comes from the architecture of the network and the use of random augmentations to enhance the diversity of generated results despite training with a limited set of inputs. As shown in Figure 11, naively using the deep U-Net architecture proposed by Zhang *et al.* [76], denoted as cGCA, results in plain overfitting on the input

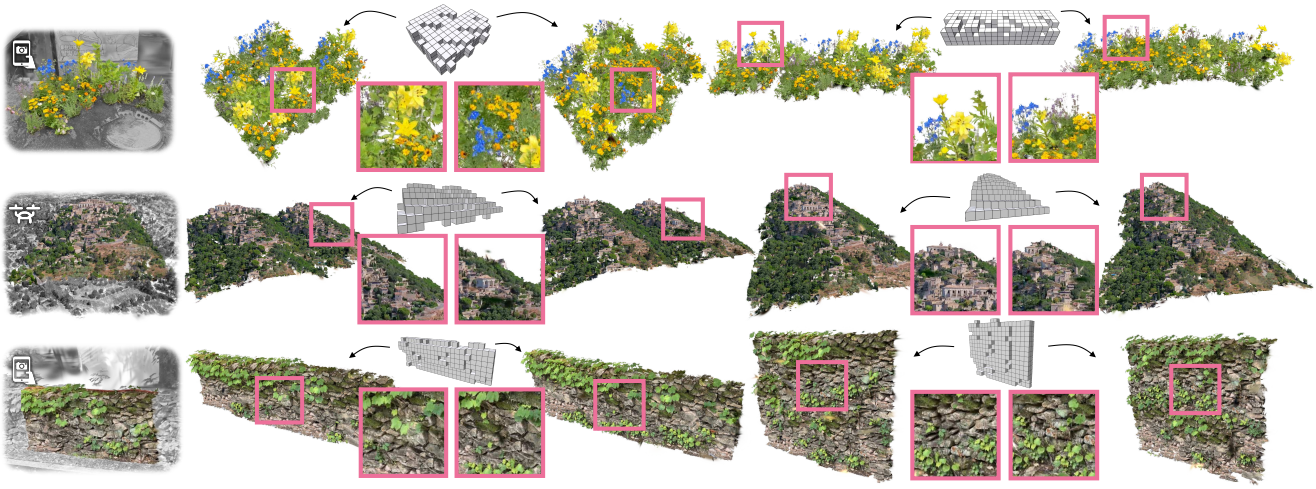


Figure 6. Starting from the exemplar on the left, we generate multiple samples for different conditioning signals. All examples are derived from real-world scenes obtained from casual mobile footage (rows 1, 3) or a drone (row 2).

Figure 7. Our method can be used to transfer the appearance of an exemplar to a mesh (row 1) or another exemplar (row 2). Given an input exemplar (column 1), and a target geometry (column 2), the target geometry is converted to conditioning voxels (column 3), and our method generates a new result (column 4).

Figure 8. Removing GCA during generation and relying solely on our sparse patch-based synthesis (column 3) leads to repetitive patterns, diminished spatial and semantic consistency, and geometry that exceeds the conditioning voxels (column 1), in contrast to our two-stage strategy with GCA (column 2).

shape and appearance. We thus introduce a smaller network with a more restricted receptive field (details in the supplemental material, Section S.2.2), which also significantly reduces model size (4.21 MB vs. 468.19 MB), training (6.25 min vs. 23.40 min for Figure 11) and inference speed (199.98 ms vs. 328.74 ms for Figure 11). Additional ablations for our patch-based consistency step are provided in the supplemental material (Section S.3).

4.4. Training and Generation Speed

A key advantage of our method is its ability to generate high-quality 3D scenes with minimal latency, enabling interactive use. Our method incurs low and fixed training overhead. As shown in Figure 9 (left), training the generative kernel from a single exemplar takes under 10 minutes, even for complex inputs. Once trained, our method supports real-time generation, enabling iterative scene design (see the supplemental video). As shown in Figure 9 (right), our two-stage pipeline (i.e., GCA generation followed by our patch-based consistency step) generates scenes in under 2 seconds, even for large scenes.

5. Limitations

Being trained on a single exemplar without external priors, our generative model struggles to extrapolate to inputs that deviate significantly from its original distribution. This limitation is illustrated in Figure 12. For the same reason, it is also crucial to carefully select the target distribution of the scene. As shown in Figure 13, properly isolating flowers from their pot allows to brush a new scene without artifacts. As generation is only conditioned through coarse voxels, ambiguities appear for complicated scenes where the same coarse geometry may be partially repeated

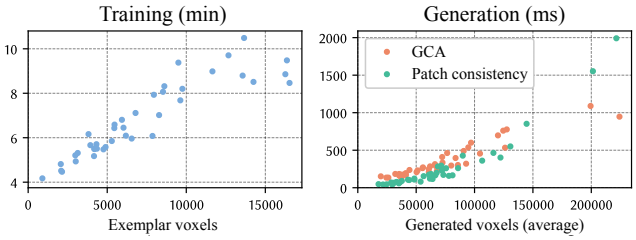


Figure 9. Timings for training (left) and generation (right). We report training times for 40 manually captured scenes covering various distributions (left). Even for a large number of voxels in the input exemplar, training time is almost always below 10 minutes. Conditioned on each manual input, we generate 10 samples and report the average generation time for GCA (red, right) and the subsequent patch consistency operation (green, right). Generation in sum takes less than 2 seconds in most cases.

throughout the scene. Since our GCA cannot observe the full generation, this results in "out of place" generated results as illustrated in Figure 14. Furthermore, after reconstructing the 3D Gaussians from voxels, no further optimization is performed, which may compromise the quality of the synthesized results, especially causing artifacts at patch boundaries as shown in Figure 15.

The resolution of the conditioning signal is prone to the trade-off between the diversity of synthesized scenes and the amount of control that can be provided to users. We provide additional analysis in the supplemental material (Section S.3.1).Future work could explore automatic resolution and scale selection or, inspired by recent multi-scale methods [51, 53], jointly train nested models and select the one matching the user's desired level of detail.

We build on the original form of 3D Gaussian Splatting [28] and our approach may benefit from recent improvements to overcome its limitations. Notably, we could integrate recent works to allow relighting of 3D Gaussians [23, 49] in particular when compositing multiple generated results. While our generative backbone is memory-efficient, 3D Gaussians may exceed 1 GB for some scenes. Recent methods could help compress them [46, 31].

6. Conclusion

We propose ExCellGen, a framework for fast, controllable, photorealistic 3D scene synthesis from a single real-world exemplar taken from in-the-wild video footage. We employ 3D Gaussian splatting for its speed, high-quality appearance, and natural composability. To amortize the generation of a complex and unstructured volume of 3D Gaussians, we introduce a two-stage approach. Each scene is first abstracted by sparse featurized voxels, which allows us to leverage Generative Cellular Automata (GCA), an efficient sparse voxel-based generative model. We achieve single-exemplar training of GCA by injecting semantically-aware features, and employing a small network trained with random augmentations within less than 10 minutes. In a second stage, we introduce a sparse patch-based consistency step to efficiently transform the coarse-generated voxels into a high-quality 3D Gaussian representation. Complete generation takes 0.5 to 2 seconds for each scene, enabling truly interactive authoring sessions. Through various examples and a fully self-contained GUI editor, we show that our method can be used to model and synthesize various distributions and enable diverse applications: controllable scene generation, appearance transfer, mixing components from various scenes, etc.

References

[1] S. Bahmani, J. J. Park, D. Paschalidou, X. Yan, G. Wetzstein, L. Guibas, and A. Tagliasacchi. Cc3d: Layout-conditioned generation of compositional 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7171–7181, 2023. 3

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 3

[3] C. Barnes and F.-L. Zhang. A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media*, 3:3–20, 2017. 3, 5

[4] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, June 2022. 3

[5] A. Bokhovkin, Q. Meng, S. Tulsiani, and A. Dai. Scenefactor: Factored latent 3d diffusion for controllable 3d scene generation. *arXiv preprint arXiv:2412.01801*, 2024. 3

[6] F. Bordes, S. Honari, and P. Vincent. Learning to generate samples from noise through infusion training. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 5

[7] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2, 4

[8] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 2

[9] Y. Chen, Y. Pan, Y. Li, T. Yao, and T. Mei. Control3d: Towards controllable text-to-3d generation. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 1148–1156, 2023. 3

[10] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5939–5948, 2019. 3

[11] J. Chung, S. Lee, H. Nam, J. Lee, and K. M. Lee. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384*, 2023. 3

[12] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann. Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 15–22, 2009. 3

[13] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 2, 3

[14] J. Deng, W. Chai, J. Guo, Q. Huang, W. Hu, J.-N. Hwang, and G. Wang. Citygen: Infinite and controllable 3d city layout generation. *arXiv preprint arXiv:2312.01508*, 2023. 3

[15] O. Diamanti, C. Barnes, S. Paris, E. Shechtman, and O. Sorkine-Hornung. Synthesis of complex image appearance from limited exemplars. *ACM Transactions on Graphics (TOG)*, 34(2):1–14, 2015. 2

[16] W. Dong, B. Yang, L. Ma, X. Liu, L. Cui, H. Bao, Y. Ma, and Z. Cui. Coin3d: Controllable and interactive 3d assets generation with proxy-guided conditioning. *arXiv preprint arXiv:2405.08054*, 2024. 3

[17] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022. 2

[18] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 571–576. 2023. 3

[19] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999. 2, 3

[20] D. Epstein, B. Poole, B. Mildenhall, A. A. Efros, and A. Holynski. Disentangled 3d scene generation with layout learning. *arXiv preprint arXiv:2402.16936*, 2024. 3

[21] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 3

[22] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM transactions on graphics (TOG)*, 23(3):652–663, 2004. 3

[23] J. Gao, C. Gu, Y. Lin, H. Zhu, X. Cao, L. Zhang, and Y. Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023. 9

[24] B. Gong, Y. Wang, X. Han, and Q. Dou. Seamlessnerf: Stitching part nerfs with gradient propagation. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–10, 2023. 3

[25] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 557–570. 2023. 3

[26] C. Jambon, B. Kerbl, G. Kopanas, S. Diolatzis, T. Leimkühler, and G. Drettakis. Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 3

[27] H. Jung, S. Nam, N. Sarafianos, S. Yoo, A. Sorkine-Hornung, and R. Ranjan. Geometry transfer for stylizing radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8565–8575, 2024. 3

[28] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. 2, 3, 4, 9

[29] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19729–19739, 2023. 7

[30] S. Kobayashi, E. Matsumoto, and V. Sitzmann. Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems*, 35:23311–23330, 2022. 4

[31] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024. 9

[32] W. Li, X. Chen, J. Wang, and B. Chen. Patch-based 3d natural scene generation from a single example. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16762–16772, 2023. 2, 3, 7

[33] Y. Liao, J. Xie, and A. Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2022. 2

[34] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023. 3

[35] F.-L. Liu, H. Fu, Y.-K. Lai, and L. Gao. Sketchdream: Sketch-based text-to-3d generation and editing. *arXiv preprint arXiv:2405.06461*, 2024. 3

[36] X. Liu, P. Tayal, J. Wang, J. Zarzar, T. Monnier, K. Tertikas, J. Duan, A. Toisoul, J. Y. Zhang, N. Neverova, et al. Uncommon objects in 3d. *arXiv preprint arXiv:2501.07574*, 2025. 2

[37] A. Lukoianov, H. Sáez de Ocáriz Borde, K. Greenewald, V. Guizilini, T. Bagautdinov, V. Sitzmann, and J. M. Solomon. Score distillation via reparametrized ddim. *Advances in Neural Information Processing Systems*, 37:26011–26044, 2024. 3

[38] N. Maruani, W. Yifan, M. Fisher, P. Alliez, and M. Desbrun. Shapeshifter: 3d variations using multiscale and sparse point-voxel diffusion, 2025. 2, 3, 4

[39] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruc-

tion in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 3

[40] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 3

[41] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 3

[42] Y. Nikankin, N. Haim, and M. Irani. Sinfusion: Training diffusion models on a single image or video. *arXiv preprint arXiv:2211.11743*, 2022. 4, 5

[43] NVIDIA, :, N. Agarwal, A. Ali, M. Bala, Y. Balaji, E. Barker, T. Cai, P. Chattopadhyay, Y. Chen, Y. Cui, Y. Ding, D. Dworakowski, J. Fan, M. Fenzi, F. Ferroni, S. Fidler, D. Fox, S. Ge, Y. Ge, J. Gu, S. Gururani, E. He, J. Huang, J. Huffman, P. Jannaty, J. Jin, S. W. Kim, G. Klár, G. Lam, S. Lan, L. Leal-Taixe, A. Li, Z. Li, C.-H. Lin, T.-Y. Lin, H. Ling, M.-Y. Liu, X. Liu, A. Luo, Q. Ma, H. Mao, K. Mo, A. Mousavian, S. Nah, S. Niverty, D. Page, D. Paschalidou, Z. Patel, L. Pavao, M. Ramezanali, F. Reda, X. Ren, V. R. N. Sabavat, E. Schmerling, S. Shi, B. Stefaniak, S. Tang, L. Tchapmi, P. Tredak, W.-C. Tseng, J. Varghese, H. Wang, H. Wang, H. Wang, T.-C. Wang, F. Wei, X. Wei, J. Z. Wu, J. Xu, W. Yang, L. Yen-Chen, X. Zeng, Y. Zeng, J. Zhang, Q. Zhang, Y. Zhang, Q. Zhao, and A. Zolkowski. Cosmos world foundation model platform for physical ai, 2025. 2

[44] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 4

[45] K. Pandey, A. Hu, C. Fuji-Tsang, O. Perel, K. Singh, and M. Shugrina. Painting with 3d gaussian splat brushes. 2025. 4

[46] P. Papantonakis, G. Kopanas, B. Kerbl, A. Lanvin, and G. Drettakis. Reducing the memory footprint of 3d gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1):1–17, 2024. 9

[47] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 3

[48] R. Po and G. Wetzstein. Compositional 3d scene generation using locally conditioned diffusion. *arXiv preprint arXiv:2303.12218*, 2023. 3

[49] Y. Poirier-Ginter, A. Gauthier, J. Philip, J.-F. Lalonde, and G. Drettakis. A Diffusion Approach to Radiance Field Relighting using Multi-Illumination Synthesis. *Computer Graphics Forum*, 2024. 9

[50] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 3

[51] Y. Rao, Y. Nie, and A. Dai. Patchcomplete: Learning multi-resolution patch priors for 3d shape completion on unseen categories. *Advances in Neural Information Processing Systems*, 35:34436–34450, 2022. 9

[52] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10901–10911, 2021. 2

[53] X. Ren, J. Huang, X. Zeng, K. Museth, S. Fidler, and F. Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. *arXiv preprint arXiv:2312.03806*, 2023. 3, 9

[54] X. Ren, Y. Lu, H. Liang, Z. Wu, H. Ling, M. Chen, S. Fidler, F. Williams, and J. Huang. Scube: Instant large-scale scene reconstruction using voxsplats. *arXiv preprint arXiv:2410.20030*, 2024. 3

[55] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 3

[56] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]). 5

[57] J. Schult, S. Tsai, L. Höllein, B. Wu, J. Wang, C.-Y. Ma, K. Li, X. Wang, F. Wimbauer, Z. He, et al. Controlroom3d: Room generation using semantic proxy rooms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6201–6210, 2024. 3

[58] T. R. Shaham, T. Dekel, and T. Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4570–4580, 2019. 4

[59] Y. Shi, P. Wang, J. Ye, M. Long, K. Li, and X. Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 3

[60] S. Szymanowicz, C. Rupprecht, and A. Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. *arXiv preprint arXiv:2312.13150*, 2023. 3

[61] S. Szymanowicz, J. Y. Zhang, P. Srinivasan, R. Gao, A. Brussee, A. Holynski, R. Martin-Brualla, J. T. Barron, and P. Henzler. Bolt3d: Generating 3d scenes in seconds. *arXiv preprint arXiv:2503.14445*, 2025. 3

[62] V. Tschernezki, I. Laina, D. Larlus, and A. Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In *2022 International Conference on 3D Vision (3DV)*, pages 443–453. IEEE, 2022. 4

[63] W. Wang, J. Bao, W. Zhou, D. Chen, D. Chen, L. Yuan, and H. Li. Sindiffusion: Learning a diffusion model from a single natural image. *arXiv preprint arXiv:2211.12445*, 2022. 4, 5

[64] Y. Wang, X. Chen, and B. Chen. Singrav: Learning a generative radiance volume from a single natural scene. *arXiv preprint arXiv:2210.01202*, 2022. 3

[65] Z. Wang, T. Wang, G. Hancke, Z. Liu, and R. W. Lau. Themestation: Generating theme-aware 3d assets from few exemplars. *arXiv preprint arXiv:2403.15383*, 2024. 3

[66] R. Wu, R. Liu, C. Vondrick, and C. Zheng. Sin3dm: Learning a diffusion model from a single 3d textured shape. *arXiv preprint arXiv:2305.15399*, 2023. 2, 4, 5

[67] R. Wu and C. Zheng. Learning to generate 3d shapes from a single example. *arXiv preprint arXiv:2208.02946*, 2022. 2, 4

[68] Z. Wu, Y. Li, H. Yan, T. Shang, W. Sun, S. Wang, R. Cui, W. Liu, H. Sato, H. Li, et al. Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation. *arXiv preprint arXiv:2401.17053*, 2024. 3

[69] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, and J. Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024. 3

[70] H. Xie, Z. Chen, F. Hong, and Z. Liu. Citydreamer: Compositional generative model of unbounded 3d cities. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9666–9675, 2024. 3

[71] K. Xie, J. Lorraine, T. Cao, J. Gao, J. Lucas, A. Torralba, S. Fidler, and X. Zeng. Latte3d: Large-scale amortized text-to-enhanced3d synthesis. *arXiv preprint arXiv:2403.15385*, 2024. 3

[72] A. Yu, G. Yang, R. Choi, Y. Ravan, J. Leonard, and P. Isola. Learning visual parkour from generated images. In *8th Annual Conference on Robot Learning*, 2024. 2

[73] H.-X. Yu, H. Duan, C. Herrmann, W. T. Freeman, and J. Wu. Wonderworld: Interactive 3d scene generation from a single image. *arXiv preprint arXiv:2406.09394*, 2024. 3

[74] Y.-J. Yuan, Y.-T. Sun, Y.-K. Lai, Y. Ma, R. Jia, and L. Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 3

[75] D. Zhang, C. Choi, J. Kim, and Y. M. Kim. Learning to generate 3d shapes with generative cellular automata. *arXiv preprint arXiv:2103.04130*, 2021. 1, 2, 5

[76] D. Zhang, C. Choi, I. Park, and Y. M. Kim. Probabilistic implicit scene completion. *arXiv preprint arXiv:2204.01264*, 2022. 5, 7, 14

[77] S. Zhou, H. Chang, S. Jiang, Z. Fan, Z. Zhu, D. Xu, P. Chari, S. You, Z. Wang, and A. Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. *arXiv preprint arXiv:2312.03203*, 2023. 4

[78] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: An interactive system for point-based surface editing. *ACM Transactions on Graphics (TOG)*, 21(3):322–329, 2002. 3

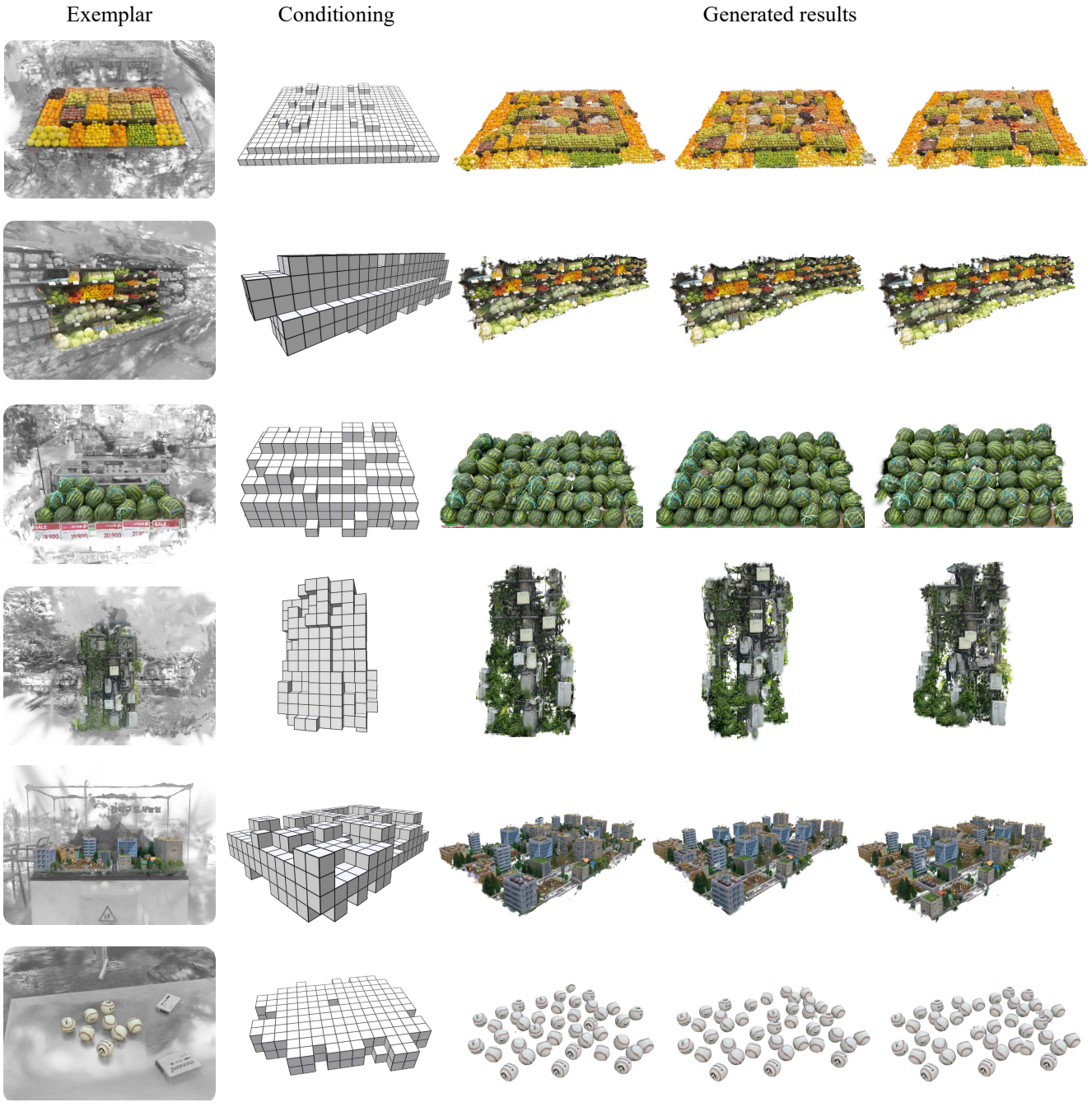


Figure 10. From a single real-world exemplar (column 1), our method can generate diverse results (columns 3-5) conditioned on coarse voxel inputs (column 2). Generation is performed and visualized within 0.5-2 seconds in our interactive editor, enabling users to iteratively refine the conditioning input to produce the target asset.

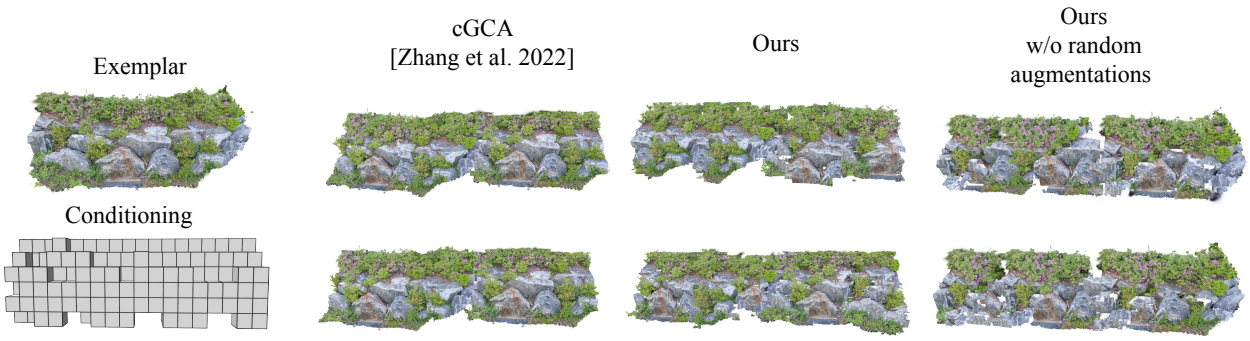


Figure 11. We propose a lightweight alternative to cGCA [76], with a reduced receptive field (see supplemental material, Section S.2.2). Given the exemplar and conditioning input (left), we generate two samples per model. cGCA tends to overfit the input exemplar, even with random augmentations. Without random augmentations, our model exhibit low sample diversity and introduce discontinuities (right-most column).

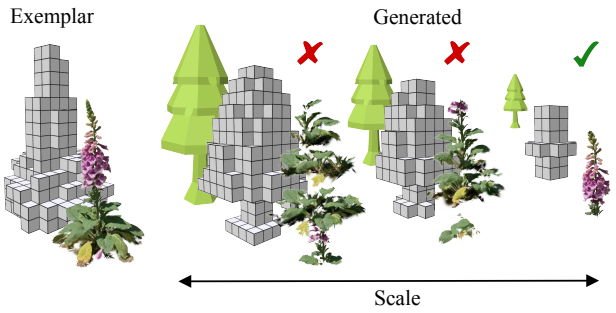


Figure 12. From a given input exemplar on the left, we generate a shape given the same input mesh at different scales. Note that the mesh is always voxelized to condition generation. If the input geometry deviates significantly from the structural distribution of the exemplar, our method struggles to produce consistent results.

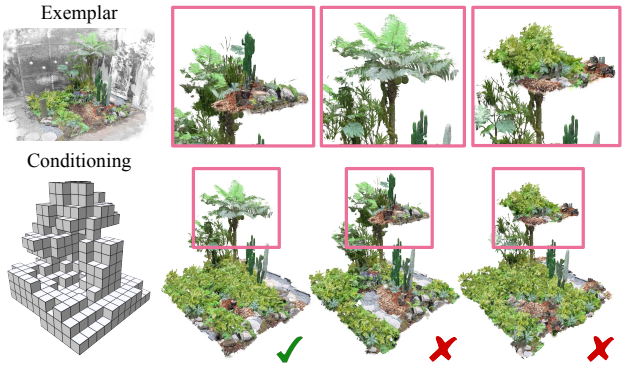


Figure 14. Due to its limited receptive field and capacity, our model can sometimes confuse distinct regions during generation, especially when semantically similar areas share the same coarse geometry.

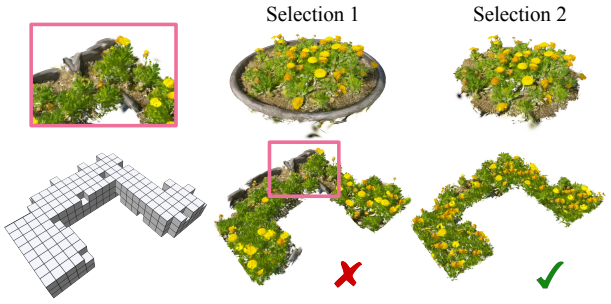


Figure 13. With two different selections of the same scene, generation can be severely hindered by structural artifacts that lie beyond the invariances that can be modeled by GCA as it operates on axis-aligned voxel grids.

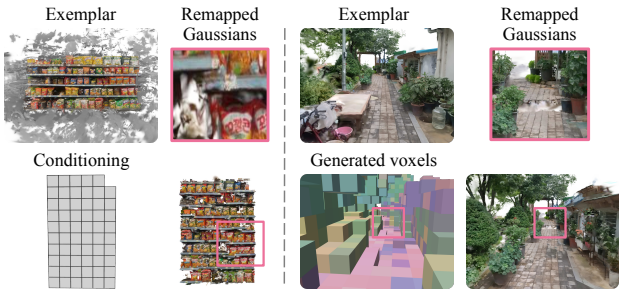


Figure 15. Our method operates on fixed-size voxels that can only be retrieved from the set of voxels in the exemplar. For axis-aligned structured scenes (left) and large scenes (right), this comes with noticeable artifacts, such as misalignments (left) and "cracks" (right).