GraphDreamer: Compositional 3D Scene Synthesis from Scene Graphs

Gege Gao[1,2,3,4] Weiyang Liu[1,5,*] Anpei Chen[2,3,4] Andreas Geiger[3,4,†] Bernhard Schölkopf[1,2,4,†]
[1]Max Planck Institute for Intelligent Systems – Tübingen [2]ETH Zürich [3]University of Tübingen
[4]Tübingen AI Center [5]University of Cambridge [*]Directional lead [†]Shared last author

graphdreamer.github.io

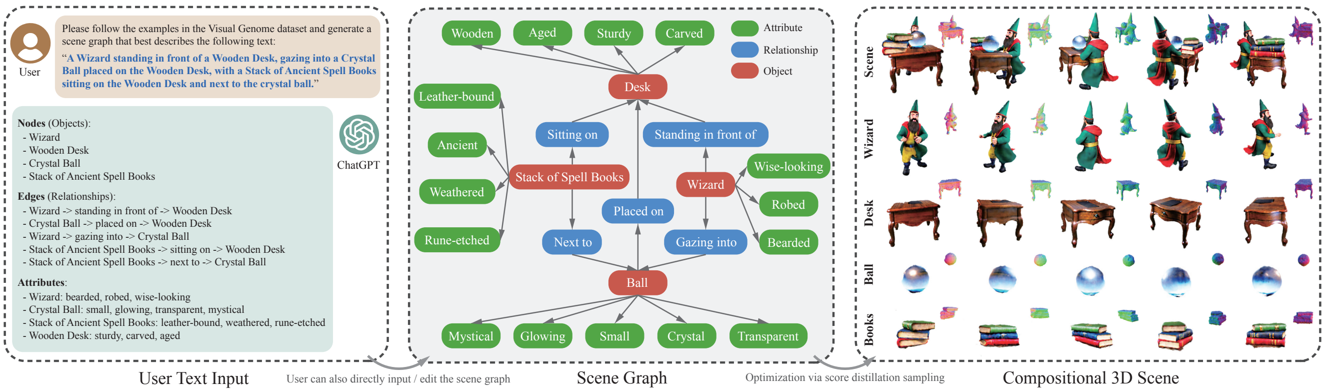

Figure 1. *GraphDreamer* takes a scene graph as input and generates a compositional 3D scene where each object is fully disentangled. To save the effort of building a scene graph from scratch, the scene graph can be generated by a language model (e.g., ChatGPT) from a user text input (left box).

Abstract

As pretrained text-to-image diffusion models become increasingly powerful, recent efforts have been made to distill knowledge from these text-to-image pretrained models for optimizing a text-guided 3D model. Most of the existing methods generate a holistic 3D model from a plain text input. This can be problematic when the text describes a complex scene with multiple objects, because the vectorized text embeddings are inherently unable to capture a complex description with multiple entities and relationships. Holistic 3D modeling of the entire scene further prevents accurate grounding of text entities and concepts. To address this limitation, we propose GraphDreamer, a novel framework to generate compositional 3D scenes from scene graphs, where objects are represented as nodes and their interactions as edges. By exploiting node and edge information in scene graphs, our method makes better use of the pretrained text-to-image diffusion model and is able to fully disentangle different objects without image-level supervision. To facilitate modeling of object-wise relationships, we use signed distance fields as representation and impose a constraint to avoid inter-penetration of objects. To avoid manual scene graph creation, we design a text prompt for ChatGPT to generate scene graphs based on text inputs. We conduct both qualitative and quantitative experiments to validate the effectiveness of GraphDreamer in generating high-fidelity compositional 3D scenes with disentangled object entities.

1. Introduction

Recent years have witnessed substantial progresses in text-to-3D generation [20, 28, 38], largely due to the rapid development made in text-to-image models [31, 32] and text-image embeddings [30]. This emerging field has attracted considerable attention due to its significant potential to revolutionize the way artists and designers work.

The central idea of current text-to-3D pipelines is to leverage knowledge of a large pretrained text-to-image generative model to optimize each randomly sampled 2D view of a 3D object such that these views resemble what the input text describes. The 3D consistency of these 2D views is typically guaranteed by a proper 3D representation (*e.g.*, neural radiance fields (NeRF) [22] in DreamFusion [28]). Despite being popular, current text-to-3D pipelines still suffer from *attribute confusion* and *guidance collapse*. Attribute confusion is a fundamental problem caused by text-image embeddings (*e.g.*, CLIP [30]). For example, models often fail at distinguishing the difference between “a black cat on a pink carpet” and “a pink cat on a black carpet”. This problem may prevent current text-to-3D generation methods from accurately grounding all attributes to corresponding objects. As the text prompt becomes even more complex, involving multiple objects, attribute confusion becomes more significant. Guidance collapse refers to the cases where the text prompt is (partially) ignored or misinterpreted by the model. This typically also happens as the text prompt gets more complex. For example, "a teddy bear pushing a shopping cart and holding baloons", with "teddy bear" being ignored. These problems largely limit the practical utility of text-to-3D generation techniques.

A straightforward solution is to model the multi-object 3D scene in a compositional way. Following this insight, recent methods [5, 17, 27, 46] condition on additional context information such as 3D layout which provides the size and location of each object in the form of non-overlapping 3D bounding boxes. While a non-overlapping 3D layout can certainly help to produce a compositional 3D scene with each object present, it injects a strong prior and greatly limits the diversity of generated scenes. The non-overlapping 3D box assumption can easily break when objects are irregular (non-cubic) and obscuring each other. For example, the text prompt "an astronaut riding a horse" can not be represented by two non-overlapping bounding boxes. To avoid these limitations while still achieving object decomposition, we propose *GraphDreamer*, which takes a scene graph (*e.g.*, [12]) as input and generates a compositional 3D scene. Unlike 3D bounding boxes, scene graphs are spatially more relaxed and can model complex object interaction. While scene graphs are generally easier to specify than spatial 3D layouts, we also design a text prompt to query ChatGPT that enables the automatic generation of a scene graph from unstructured text. See Figure 1 for an illustrative example.

GraphDreamer is guided by the insight that a scene graph can be decomposed into a separate and semantically unambiguous text description of every node and edge[1]. The decomposition of a scene graph into multiple textual descriptions makes it possible to distill knowledge from text-to-image diffusion models, similar to common text-to-3D methods. Specifically, to allow each object to be disentangled from the other objects in the scene, we use separate identity-aware positional encoder networks (*i.e.*, object feature fields) to encode object-level semantic information and a shared *Signed Distance Field* (SDF) network to decode the SDF value from identity-aware positional features. The color value is decoded in a way similar to the SDF value. Scene-level rendering is performed by integrating objects based on the smallest SDF value at each sampled point in 3D space. More importantly, with both SDF and color values of each object, we propose an identity-aware object rendering that, in addition to a global rendering of the entire 3D scene, renders different objects separately. Our local identity-aware rendering allows the gradient from the text-dependent distillation loss (*e.g.*, score distillation sampling [28]) to be back-propagated selectively to corresponding objects without affecting the other objects. The overall 3D scene will be simultaneously optimized with the global text description to match the scene semantics to the global text. In summary, we make the following contributions:

- To the best of our knowledge, GraphDreamer is the first 3D generation method that can synthesize compositional 3D scenes from either scene graphs or unstructured text descriptions. No 3D bounding boxes are required as input.
- GraphDreamer uses scene graphs to construct a disentangled representation where each object is optimized via its related text description, avoiding object-level ambiguity.
- GraphDreamer is able to produce high-fidelity complex 3D scenes with disentangled objects, outperforming both state-of-the-art text-to-3D methods and existing 3D-bounding-box-based compositional text-to-3D methods.
- In Appendix **??**, we envision a new paradigm of semantic 3D reconstruction – *Inverse Semantics*, where a vision-language model (*e.g.*, GPT4-V) is used to extract a scene graph from an input image (*i.e.*, scene graph encoder) and GraphDreamer is used to generate a compositional 3D scene from the scene graph (*i.e.*, scene graph decoder).

2. Related Work

Text-to-2D generation. Driven by large-scale image-text aligned datasets [34], text-to-image generation models [1, 31, 32] have made great progress in producing highly realistic images. Among these models, generative diffusion models learn to gradually transform a noisy latent z with noise ϵ typically from a Gaussian distribution, towards image data x that reproduce the semantics of a given text prompt y. This generative process slowly adds structure to the noise, based on a weighted denoising score matching objective [11, 26].

2D-lifting for 3D generation. In contrast to existing text-to-image generation models, text-guided 3D generative models [9, 20, 21, 23, 25, 28, 37, 38] usually optimize a 3D model by guiding its randomly rendered 2D view based on the pretraining knowledge of some text-to-image generation model, because of the shortage of text-3D paired assets. DreamFusion [28] and subsequent work [4, 16, 36, 41, 43] propose to optimize the 3D model by distilling a pretrained diffusion model [31, 32] via score distillation sampling [38].

Generate objects with SDF. In text-to-3D generation, recent works [9, 16, 18, 20, 28, 36, 38, 39, 41] parameterized the 3D scene as a NeRF [2, 22] or a hybrid pipeline combining NeRFs with a mesh refiner [10, 16, 35, 37]. In our approach, we use a signed distance field (SDF) as the geometry representation instead of NeRF densities, as we aim at modeling multi-object scenes in a compositional way, where objects may be coupled in various ways. SDF provides cru-

[1] Cf. the assumption of *independent causal mechanisms* [33]

cial inside/outside information, allowing for geometry constraints to prevent unexpected intersections between objects, and is ideal for complex scenes as it facilitates customization of initial locations and scales of object SDFs.

Hybrid 3D representation for disentanglement. Another line of work uses hybrid representations to learn disentangled 3D objects [14, 19, 42]. The works [6, 7] put forward a hybrid approach that represents the face/body as meshes and the hair/clothing as NeRFs, enabling a disentangled reconstruction of avatars. [47] adopts this representation and proposes a text-to-3D method that generates compositional head avatars. However, the use of a parametric head model limits this method to human head generation. Their disentanglement only applies to two objects (*e.g.*, face and hair), and in contrast, ours can be used for multiple objects.

3. Preliminaries

Score distillation sampling (SDS). SDS [28] is a technique that optimizes a 3D model by distilling a pretrained text-to-image diffusion model. Given a noisy image z_t rendered from a 3D model parameterized by Θ, and a pretrained text-to-image diffusion model with a learned noise prediction network $\epsilon_\phi(\cdot)$, SDS uses $\epsilon_\phi(\cdot)$ as a score function that predicts the sampled noise ϵ contained in x_t at noise level t as $\hat{\epsilon}_\phi(y,t) = \epsilon_\phi(z_t; y, t)$, where y is a given conditional text embedding. The score is then used to warp the noisy z_t towards real image distributions, by guiding the direction of the gradients that update the parameters Θ of the 3D model:

$$\nabla_\Theta \mathcal{L}(z; y) = \mathbb{E}_{t,\epsilon}\left[w(t)\left(\hat{\epsilon}_\phi(y,t) - \epsilon\right)\frac{\partial z}{\partial \Theta}\right] \tag{1}$$

where $w(t)$ is a weighting function that depends on t and ϵ is the sampled isotropic Gaussian noise, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

SDF volume rendering. To render a pixel color of a target camera, we cast a ray $\mathbf{r}$ from the camera center $\mathbf{o}$ along its viewing direction $\mathbf{d}$, then sample a series of points $\mathbf{p} = \mathbf{o} + t\mathbf{d}$ in between the near and far intervals $[t_n, t_f]$. Following NeRF [22], the ray color $C(\mathbf{r})$ can be approximated by integrating the point samples,

$$C(\mathbf{r}) = \sum_{i=1}^{N} w_i c_i = \sum_{i=1}^{N} T_i \alpha_i c_i \tag{2}$$

where w_i is the color weighting function, T_i represents the cumulative transmittance, which is calculated as $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$, and α_i denotes the piece-wise constant opacity value for the i-th sub-interval, with $\alpha_i \in [0,1]$.

Unlike NeRFs that directly predict the density for a given position $\mathbf{p}$, methods based on implicit surface representation learn to map $\mathbf{p}$ to a signed distance value with a trainable network and extract the density [45] or opacity [40] from the SDF with a deterministic transformation. We extract the opacity following NeuS [40]. NeuS formulates the transformation function based on an unbiased weighting function, which ensures that the pixel color is dominated by the intersection point of camera ray with the zero-level set of SDF,

$$\alpha_i = \max\left(\frac{\Phi_\beta(u_i) - \Phi_\beta(u_{i+1})}{\Phi_\beta(u_i)}, 0\right) \tag{3}$$

where $\Phi_\beta(\cdot)$ is the Sigmoid function with a trainable steepness β, and u_i is the SDF value of the sampled position.

4. Method

Consider generating a scene of M objects, $\mathcal{O} = \{o_i\}_{i=1}^M$ from a global text prompt y^g. When the scene is complex or has many attributes and inter-object relationships to specify, y^g will become very long, and the generation will be accompanied by guidance collapse [3, 15]. We thus propose to first generate a **scene graph** $\mathcal{G}(\mathcal{O})$ from y^g following the setting of [12], which precisely describes object attributes and inter-object relationships. We provide an example of a four-object scene in Figure 1 for better illustration.

4.1. Leveraging Scene Graphs for Text Grounding

Given user text input y^g, objects $\{o_i\}_{i=1}^M$ in the text (which can be detected either manually or automatically, *e.g.*, using ChatGPT[2]) form the **nodes** in graph $\mathcal{G}(\mathcal{O})$, as shown in Figure 1. To provide more details to an object o_i, the user can add additional descriptions, such as "Wise-looking" and "Leather-bound", which become the **attributes** attached to o_i in $\mathcal{G}(\mathcal{O})$. Combining o_i with all its attributes simply by commas, we get an **object prompt** $y^{(i)}$ for o_i that can be processed by text encoders.

The relationship between each pair of objects o_i and o_j is transformed into **edge** $e_{i,j}$ in $\mathcal{G}(\mathcal{O})$. For instance, the edge between node "Wizard" and "Desk" is "Standing in front of". For a graph with M nodes, there are possibly C_2^M edges. By combining o_i, $e_{i,j}$, and o_j, we obtain **edge prompt** $y^{(i,j)}$ that exactly defines the pairwise relationship, *e.g.*, "Wizard standing in front of Wooden Desk". Note that there might be no edge between two nodes, *e.g.*, between "Wizard" and "Stack of Ancient Spell Books". We denote the number of existing edges in $\mathcal{G}(\mathcal{O})$ as K, with $K \le C_2^M$.

From this example, we also see that using graph $\mathcal{G}(\mathcal{O})$ is a better way to customize a scene compared to a pure text description y^g, in terms of both flexibility in attaching attributes to objects and accuracy in defining relationships. By processing the input scene graph, we now obtain a set of $(1+M+K)$ prompts $\mathcal{Y}(\mathcal{G}(\mathcal{O}))$ as:

$$\mathcal{Y}(\mathcal{G}(\mathcal{O})) = \{y^g, y^{(i)}, y^{(i,j)} \mid o_i \in \mathcal{O}, e_{i,j} \in \mathcal{G}(\mathcal{O})\} \tag{4}$$

which are used to guide scene generation from the perspective of both individual objects and pairwise relationships.

[2]ChatGPT4, https://chat.openai.com

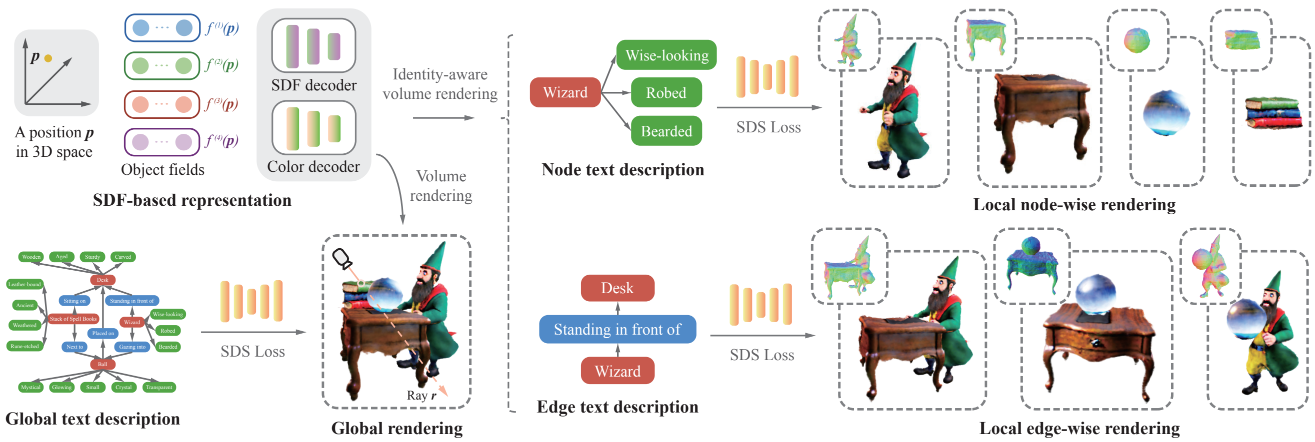

Figure 2. The overall pipeline of GraphDreamer. Specifically, GraphDreamer first decomposes the scene graph into global, node-wise and edge-wise text description, and then optimizes the SDF-based objects in the 3D scene using their corresponding text description.

GraphDreamer consists of three learnable modules: a positional feature encoder $\mathcal{F}_\theta(\cdot)$, a signed distance network $u_{\phi_1}(\cdot)$, and a radiance network $c_{\phi_2}(\cdot)$. The entire model is parameterized by $\Theta = \{\theta, \phi_1, \phi_2\}$. There are **two goals** in optimizing GraphDreamer: (i) to model the complete geometry and appearance of each object, and (ii) to ensure that object attributes and interrelationships are in accordance with the scene graph $\mathcal{G}(\mathcal{O})$. The overall training process is illustrated in Figure 2.

4.2. Disentangled Modeling of Objects

Positional encodings are useful for networks to identify the location it is currently processing. To achieve the first goal of making objects separable, we need to additionally identify which object a position belongs to. Therefore, instead of one positional feature embedding, we encode a position $\mathbf{p}$ into multiple feature embeddings by introducing a set of positional hash feature encoders, each parameterized by θ_i, corresponding to the number of objects,

$$\mathcal{F}_\theta(\cdot) = \{\mathcal{F}_{\theta_i}(\cdot)\}_{i=1}^M \quad \theta = \{\theta_i\}_{i=1}^M \tag{5}$$

These feature encoders then form different object fields, *i.e.*, one field per object across the same scene space $\Omega \subset \mathbb{R}^3$.

Individualized object fields. Given a position $\mathbf{p} \in \Omega$, the feature that forms the field of object o_i is obtained as:

$$f^{(i)}(\mathbf{p}) = \mathcal{F}_{\theta_i}(\mathbf{p}) \in \mathbb{R}^F \quad i \in \{1, \cdots, M\} \tag{6}$$

where F is the number of feature dimensions, the same for all $\mathcal{F}_{\theta_i}(\cdot)$. Here, for each $\mathcal{F}_{\theta_i}(\cdot)$ we adopt the multi-resolution hash grid encoding from Instant NGP [24] following [16, 36, 41] to reduce computational cost.

These identity-aware feature embeddings are then passed to the shared shallow MLPs for SDF and color prediction, *e.g.*, the SDF $u^{(i)}(\mathbf{p}) \in \mathbb{R}$ and color $c^{(i)}(\mathbf{p}) \in \mathbb{R}^3$ values for object o_i's field are predicted as:

$$u^{(i)}(\mathbf{p}) = u_{\phi_1}\big(f^{(i)}(\mathbf{p})\big) \qquad c^{(i)}(\mathbf{p}) = c_{\phi_2}\big(f^{(i)}(\mathbf{p})\big) \tag{7}$$

where $u^{(i)}(\mathbf{p})$ indicates the signed distance value from position $\mathbf{p}$ to the closest surface of object o_i, with negative values inside o_i and positive values outside, and $c^{(i)}(\mathbf{p})$ the color value in o_i's field where only object o_i is considered. Here, we follow prior work [44] to initialize the SDF approximately as a sphere. We transform $u^{(i)}(\mathbf{p})$ into opacity with Eq. (3) as $\gamma^{(i)}(\mathbf{p})$ for the volume rendering of object fields.

In scenes where mutual-obscuring relationships are involved, to generate the complete geometry and appearance for each object, we need to make hidden object surfaces visible. Therefore, the scene needs to be properly decomposed before rendering the objects.

Scene space decomposition. Intuitively, a position $\mathbf{p} \in \Omega$ can be occupied by at most one object. Since the SDF determines the boundary of an object, $\mathbf{p}$ can thus be identified as belonging to object o_i if its SDF values $\{u^{(i)}\}_{i=1}^M$ are minimized at index i. Based on this, we define an one-hot identity (column) vector $\boldsymbol{\lambda}(\mathbf{p})$ for each position $\mathbf{p}$ as:

$$\boldsymbol{\lambda}(\mathbf{p}) = \underset{i=1,\cdots,M}{\operatorname{argmax}} \left\{-u^{(i)}(\mathbf{p})\right\} \in \{0,1\}^M \tag{8}$$

Based on $\boldsymbol{\lambda}(\mathbf{p})$, we can decompose the scene into identity-aware sub-spaces and render each object individually with all other objects removed.

Identity-aware object rendering. To render object o_i, given a position $\mathbf{p}$, we multiply its opacity $\gamma^{(i)}$ in o_i's field with $\lambda^{(i)}(\mathbf{p})$ to obtain the opacity for only object o_i as:

$$\alpha^{(i)}(\mathbf{p}) = \lambda^{(i)}(\mathbf{p}) \cdot \gamma^{(i)} \in [0, +\infty) \tag{9}$$

where $\lambda^{(i)}(\mathbf{p})$ is the i-th element in vector $\boldsymbol{\lambda}(\mathbf{p})$. If $\lambda^{(i)}(\mathbf{p}) = 1$, which means $\mathbf{p}$ is identified as most likely to be occupied by object o_i, we have opacity $\alpha^{(i)}(\mathbf{p}) \geq 0$ in object o's field only, while in all other object fields $\mathbf{p}$ will be empty. Based on this identity-aware opacity $\alpha^{(i)}(\mathbf{p})$, we can obtain the ray color with object o_i present only as:

$$C_{\mathbf{r}}^{(i)} = \sum_{\mathbf{p}} \alpha^{(i)}(\mathbf{p}) T^{(i)}(\mathbf{p}) \cdot c^{(i)}(\mathbf{p}) \tag{10}$$

where the cumulative transmittance $T^{(i)}(\mathbf{p}_j)$ is defined following Eq. (2). By aggregating all rendered pixels, an object image $C^{(i)}$ is obtained, which contains object o_i only. With $C^{(i)}$ and the object prompt $y^{(i)}$ from the scene graph $\mathcal{G}(\mathcal{O})$, we can thus define the object SDS loss following Eq. (1) as:

$$\nabla_\Theta \mathcal{L}^{(i)}\big(C^{(i)}; y^{(i)}\big) \quad o_i \in \mathcal{O} \tag{11}$$

to supervise GraphDreamer at the object level.

4.3. Pairwise Modeling of Object Relationships

By building up a set of identity-aware object fields, we are now able to render objects in $\mathcal{G}(\mathcal{O})$ individually to match $y^{(i)}$. To make two related objects o_i and o_j respect the relationship defined in edge prompt $y^{(i,j)}$, we need to render o_i and o_j jointly. Therefore, a combination of two object fields is required.

Edge rendering. Given an edge $e_{i,j}$ connecting nodes o_i and o_j, we can accumulate an edge-wise opacity $\alpha^{(i,j)}(\mathbf{p})$ at position $\mathbf{p}$ from opacity values of o_i and o_j based on the one-hot identity vector $\boldsymbol{\lambda}(\mathbf{p})$ as:

$$\alpha^{(i,j)}(\mathbf{p}) = \sum_{k \in \{i,j\}} \lambda^{(k)}(\mathbf{p}) \cdot \gamma^{(k)}(\mathbf{p}) \tag{12}$$

and similarly, an edge-wise color $c^{(i,j)}(\mathbf{p})$ at position $\mathbf{p}$ as:

$$c^{(i,j)}(\mathbf{p}) = \sum_{k \in \{i,j\}} \lambda^{(k)}(\mathbf{p}) \cdot c^{(k)}(\mathbf{p}) \tag{13}$$

from which we can render a ray color across two object fields of o_i and o_j following the same integration process as Eq. (2), and obtain an edge image $C^{(i,j)}$ in with both objects involved. We can thus define an edge SDS loss as:

$$\nabla_\Theta \mathcal{L}^{(i,j)}\big(C^{(i,j)}; y^{(i,j)}\big) \quad e_{i,j} \in \mathcal{G}(\mathcal{O}) \tag{14}$$

to match the edge prompt $y^{(i,j)}$.

Scene rendering. To provide global scene graph $\mathcal{G}(\mathcal{O})$ constraints, we further render the whole scene globally by combining all object fields together. Similarly as for the edge rendering, we use $\boldsymbol{\lambda}(\mathbf{p})$ to accumulated the global opacity $\alpha^g(\mathbf{p})$ and a global color $c(\mathbf{p})$ at position $\mathbf{p}$ over all object fields as:

$$\alpha^g(\mathbf{p}) = \boldsymbol{\lambda}^T(\mathbf{p}) \cdot \boldsymbol{\gamma}(\mathbf{p}) \qquad c^g(\mathbf{p}) = \boldsymbol{\lambda}^T(\mathbf{p}) \cdot \boldsymbol{c}(\mathbf{p}) \tag{15}$$

with $\boldsymbol{\gamma}(\mathbf{p})$ and $\mathbf{c}(\mathbf{p})$ denote the column vectors of opacity and color values at $\mathbf{p}$ over all object fields. Through the integration process, we can render the entire scene into a scene image C^g that represents the entire scene graph. We define a scene-level SDS loss, $\nabla_\Theta \mathcal{L}^g\big(C^g; y^g\big)$, to globally optimize GraphDreamer to match the scene prompt y^g.

Efficient SDS guidance. So far, a number of $(M + K + 1)$ SDS losses are introduced, corresponding to the number of prompts we obtained in Eq. (4). However, optimizing all constraints jointly is intractable. Instead, in each training step, we include two SDS losses only: **(i)** an object SDS loss Eq. (11) for only one object o_i, with each step choosing a different object o_i looping through $\mathcal{O}$; **(ii)** an edge SDS loss Eq. (14) of one $e_{i,j}$ connected to o_i, looping through all existing edges connected to o_i. The scene SDS loss is included only in the training step after each traversal of $\mathcal{O}$, and is used solely in that step without any other SDS loss. Thus, the total SDS loss for optimizing GraphDreamer is:

$$\mathcal{L}_{SDS} = \begin{cases} \nabla_\Theta \mathcal{L}^{(i)} + \nabla_\Theta \mathcal{L}^e, & i = s \,\%\, (M+1) \\ \nabla_\Theta \mathcal{L}^g, & \text{others} \end{cases} \tag{16}$$

where s indexes the current training step, and e refers to one of the edges connected with o_i.

4.4. Training Objectives

Apart from the SDS guidance, to further optimize the prediction of unobserved positions, *i.e.*, inside objects and on hidden surfaces at object intersections, we include two geometry constraints for physically plausible shape completion.

Penetration constraint. Since each point $\mathbf{p} \in \Omega$ can be inside or on the surface of at most one object o in the set of objects $\mathcal{O}$, and for regions outside the actual objects, $u(\mathbf{p}) \in (0, +\infty)$ with $\mathbf{p} \in \Omega \setminus \mathcal{O}$, we define a penetration measurement at point p:

$$\mathcal{N}^-(\mathbf{p}) = \sum_{i=1}^{M} \max\Big\{\mathrm{sgn}\big(-u^{(i)}(\mathbf{p})\big), 0\Big\} \tag{17}$$

where $\mathrm{sgn}(x)$ is the sign function. Intuitively, $\mathcal{N}^-(\mathbf{p})$ measures the number of objects inside which the point $\mathbf{p}$ is located, according to the predicted $u^{(i)}(\mathbf{p})$. Using this measurement, we propose to implement a penetration constraint,

$$\mathcal{L}_{penet}(\mathbf{p}) = \max\big\{0, \mathcal{N}^-(\mathbf{p}) - 1\big\} \tag{18}$$

to constrains the penetration number $\mathcal{N}^-(\mathbf{p})$ not to exceed 1, which is averaged over all sampled points during training.

Eikonal constraint. At each sampled position $\mathbf{p}$, we adopt the Eikonal loss [8] on SDF values $u^{(i)}(\mathbf{p})$ from all object fields, formulated as

$$\mathcal{L}_{eknl} = \frac{1}{NM} \sum_{i,p} \left(\left\| \nabla u^{(i)}(\mathbf{p}) \right\|_2 - 1 \right) \tag{19}$$

where N is the size of the sample set $\mathcal{P}_\mathbf{r}$ on ray $\mathbf{r}$.

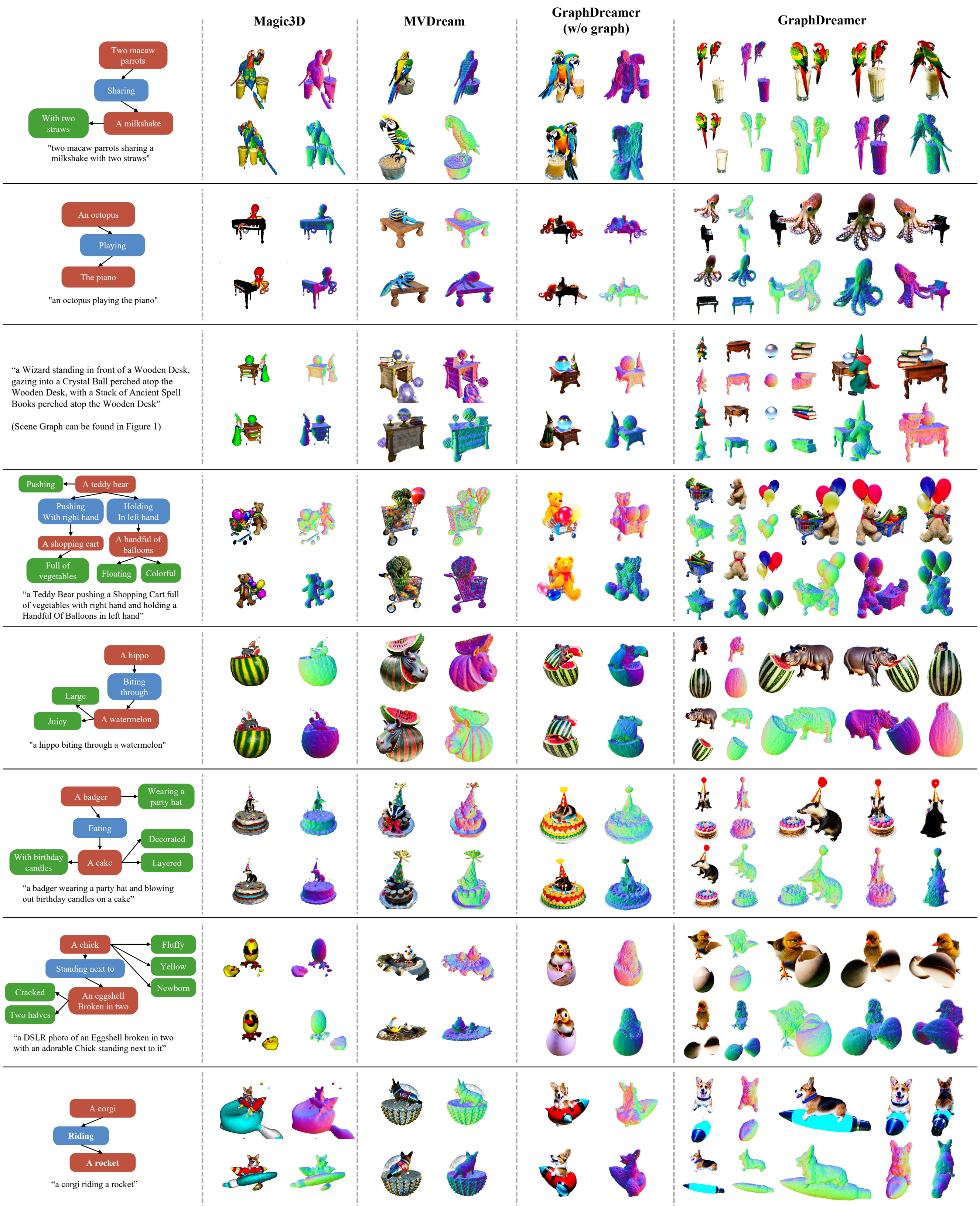

Figure 3. Qualitative comparison with baseline approaches and the ablated configuration (w/o graph). GraphDreamer generates scenes with all composing objects being separable. Moreover, with accurate guidance from scene graphs, object attributes and inter-object relationships produced by GraphDreamer match the given prompts better. We recommend to zoom-in for details.

CLIP Score	Magic3D [16]	MVDream [36]	GraphDreamer (w/o graph)	GraphDreamer
mean	0.3267	0.3102	0.3019	**0.3326**
std.	0.0362	0.0061	0.0254	0.0252

Table 1. Quantitative results. The mean and standard deviation (std.) values are summarized from CLIP scores of 30 multi-object scenes, with the number of objects in the scene ≥ 2. For better comparison, we provide the result of GraphDreamer (w/o graph), the configuration with the scene graph $\mathcal{G}(\mathcal{O})$ dropped from GraphDreamer, and thus the conditioning text for all renderings collapse to a single y^g.

Total training loss. Our final loss function for training GraphDreamer thus consists of four terms:

$$\mathcal{L}_\Theta = \beta_1 \mathcal{L}_{SDS} + \beta_2 \mathcal{L}_{penet} + \beta_3 \mathcal{L}_{eknl} \quad (20)$$

where $\{\beta_1, \beta_2, \beta_3\}$ are hyperparameters.

5. Experiments and Results

Implementation details. We adopt a two-stage coarse-to-fine optimization strategy following previous work [4, 16, 28, 36]. In the first stage of 10K denoising steps, we render images of 64×64 resolution only for faster convergence and use DeepFloyd-IF[3] [13, 32] as our guidance model, which is also trained to generate 64px images. In the second stage of 10K steps, the model is refined by rendering 256px images and uses Stable Diffusion[4] [31] as guidance. Both stages of optimization use 1 Nvidia Quadro RTX 6000 GPU; GraphDreamer uses 16.88/18.58/20.05 GB for generating 2/3/4 objects, while Magic3D/MVDream uses 11.44/20.33 GB.

Baseline approaches. We report results of two state-of-the-art approaches, Magic3D [16] and MVDream [36]. Both approaches use a frozen guidance model without additional learnable module [41] and do not have special initialization requirements [4] for geometry. We use the same guidance models and strategy to train Magic3D, while for MVDream, since it proposes to use a fine-tuned multi-view diffusion model[5], we follow its official training protocol and use its released diffusion model as guidance. The experimental results of GraphDreamer and the baselines are all obtained after training for 20K steps in total.

Evaluation criteria. We report the CLIP Score [29] in quantitative comparison with baseline models and evaluation on object decomposition. The metric is defined as:

$$\text{CLIPScore}(C, y) = \cos \langle E_C(C), E_Y(y) \rangle \quad (21)$$

which measures the similarity between a prompt text y for an image C and the actual content of the image, with $E_C(C)$ the visual CLIP embedding and $E_Y(y)$ the textual CLIP embedding, both encoded by the same CLIP model[6].

[3]DeepFloyd-IF, huggingface.co/DeepFloyd/IF-I-XL-v1.0
[4]Stable diffusion, huggingface.co/stabilityai/stable-diffusion-2-1-base
[5]MVDream-sd-v2.1-base-4view, huggingface.co/MVDream/MVDream
[6]CLIP B/32, huggingface.co/openai/clip-vit-base-patch32

CLIP Score	w. Self Prompt ↑		w. Other Prompts ↓	
	mean	std.	mean	std.
GraphDreamer	0.308	0.012	0.201	0.009

Table 2. The CLIP scores of individual object images $C^{(i)}$. Metric **w. Self Prompt** refers to scores calculated between $C^{(i)}$ and its own prompt $y^{(i)}$, and **w. Other Prompts** between $C^{(i)}$ and prompts of all other objects in the same scene $\{y^{(j)}, j \neq i, o_j \in \mathcal{O}\}$. Detailed experimental settings and analysis on these figures as well as the on the chart showing in Figure 4, can be found in Subsection 5.3.

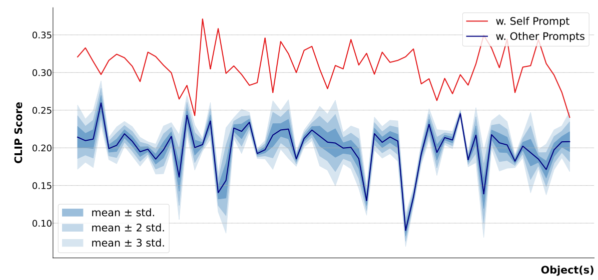

Figure 4. Error bands of object CLIP scores.

5.1. Comparison to Other Methods

We report quantitative results in Table 1 and qualitative results in Figure 3. The figures in Table 1 are summarized from CLIP scores of 30 multi-object scenes, with the number of objects ≥ 2. GraphDreamer (full model) achieves the highest CLIP score. Qualitative results shown in Figure 3 also suggest that GraphDreamer is generally more applicable in producing multiple objects in a scene with various inter-object relationships. As can be observed from baseline results, semantic mismatches are more commonly found, such as, in the example of "two macaw parrots sharing a milkshake with two straws", Magic3D generates two "milkshakes" and MVDream produces one "parrot" only, which mixed up the number attribute of "milkshake" and "parrots", and in the example of "a hippo biting through a watermelon", two objects "hippo" and "watermelon" are blended into one object; GraphDreamer, on the other hand, models both individual attributes and inter-object relationships correctly based on the guidance of the input scene graph.

5.2. Ablation Study

To evaluate the effect of introducing scene graph $\mathcal{G}(\mathcal{O})$ as guidance on preventing guidance collapse, we consider to drop $\mathcal{G}(\mathcal{O})$ from GraphDreamer, termed **w/o graph**, and compare the results in Table 1 and Figure 3. GraphDreamer (w/o graph) collapses the conditioning text from a set of prompts defined in Eq. (4) to only the global prompt y^g and thus involves no object/edge rendering (described in Section 4.2 and 4.3) in training; L_{SDS} from Eq. (16) becomes $\nabla_\Theta \mathcal{L}^g$, with all other implementation details unchanged. As reported in the fourth column of Table 1, the mean CLIP score of the ablated configuration decreases by more than 3

Figure 5. More qualitative compositional examples from GraphDreamer.

std. compared to the full model, indicating that the ability of this configuration to generate 3D scenes that match given prompts is significantly reduced. The results in Figure 3 also corroborate such a decline, given that the problems such as missing objects and attribute confusion arise again. Both evaluations suggest the need for the scene graph $\mathcal{G}(\mathcal{O})$ in GraphDreamer for combating guidance collapse problems.

5.3. Decomposition Analysis

We consider scenes with the number of objects ≥ 2. To further quantitatively evaluate whether objects in a scene are well separated and rendered into object images individually, we calculate two CLIP metrics for each object image $C^{(i)}$: **(i) w. Self Prompt** (abbr., wSP) refers to the CLIP score between $C^{(i)}$ and its own object prompt $y^{(i)}$, and **(ii) w. Other Prompts** (abbr., wOP) refers to the CLIP scores between $C^{(i)}$ and all other object prompts $\{y^{(j)}, j \neq i, o_j \in \mathcal{O}\}$ in the same scene. Intuitively, if a scene is well decomposed, each object image $C^{(i)}$ should contain one object o_i only without any part of other objects, and thus the scores wSP should be much higher than the scores wOP. Table 2 reports a statistical summary on the metrics of 64 objects, for each object, we render images from 4 orthogonal views $C_v^{(i)}$ ($v = 1, 2, 3, 4$), and thus we get 4 wSP scores and multiple wOP scores per object. We calculate the mean and standard deviation (std.) of these wSP and wOP scores separately over the view images. The figures reported in the table are averaged over all 64 objects. The mean and std. values are also presented in an error band graph in Figure 4, where the x-axis is the index of objects. From this graph it can be observed more obviously that the wSP CLIP score is significantly higher than the wOP CLIP score, without overlap between the mean wSP scores and the mean $\pm$ 3 std. error band of the wOP scores, which shows that the scenes are properly decomposed into individual objects. More compositional examples can be found in Figure 5.

6. Concluding Remarks

This paper proposes GraphDreamer, which generates compositional 3D scenes from scene graphs or text (by leveraging GPT4-V to produce a scene graph from the text). GraphDreamer first decomposes the scene graph into global, node-wise, and edge-wise text descriptions and then optimizes the SDF-based objects with the SDS loss from their corresponding text descriptions. We conducted extensive experiments to show that GraphDreamer is able to prevent attribute confusion and guidance collapse, generating disentangled objects.

References

[1] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers, 2023.

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021.

[3] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *ACM Trans. on Graphics*, 2023.

[4] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3D: Disentangling geometry and appearance for high-quality text-to-3D content creation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023.

[5] Helisa Dhamo, Fabian Manhardt, Nassir Navab, and Federico Tombari. Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021.

[6] Yao Feng, Jinlong Yang, Marc Pollefeys, Michael J Black, and Timo Bolkart. Capturing and animation of body and clothing from monocular video. In *SIGGRAPH Asia*, 2022.

[7] Yao Feng, Weiyang Liu, Timo Bolkart, Jinlong Yang, Marc Pollefeys, and Michael J Black. Learning disentangled avatars with hybrid 3D representations. *arXiv.org*, 2309.06441, 2023.

[8] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proc. of the International Conf. on Machine learning (ICML)*, 2020.

[9] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[10] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields, 2022.

[11] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *arXiv.org*, 2107.00630, 2023.

[12] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV)*, 2017.

[13] DeepFloyd Lab. *DeepFloyd*, 2023.

[14] Gengyan Li, Abhimitra Meka, Franziska Mueller, Marcel C Buehler, Otmar Hilliges, and Thabo Beeler. Eyenerf: a hybrid representation for photorealistic synthesis, animation and relighting of human eyes. *ACM Trans. on Graphics*, 2022.

[15] Yumeng Li, Margret Keuper, Dan Zhang, and Anna Khoreva. Divide & bind your attention for improved generative semantic nursing. *arXiv.org*, 2307.10864, 2023.

[16] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[17] Yiqi Lin, Haotian Bai, Sijia Li, Haonan Lu, Xiaodong Lin, Hui Xiong, and Lin Wang. Componerf: Text-guided multi-object compositional nerf with editable 3D scene layout. *arXiv.org*, 2303.13843, 2023.

[18] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *ICCV*, 2023.

[19] Weiyang Liu, Zhen Liu, Liam Paull, Adrian Weller, and Bernhard Schölkopf. Structural causal 3d reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[20] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3D shapes and textures. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[21] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[22] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[23] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia*, 2022.

[24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. on Graphics*, 2022.

[25] Gimin Nam, Mariem Khlifi, Andrew Rodriguez, Alberto Tono, Linqi Zhou, and Paul Guerrero. 3d-ldm: Neural implicit 3d shape generation with latent diffusion models, 2022.

[26] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv.org*, 2021.

[27] Ryan Po and Gordon Wetzstein. Compositional 3D scene generation using locally conditioned diffusion. *arXiv.org*, 2303.12218, 2023.

[28] Ben Poole, Ajay Jain, Jonathan Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv.org*, 2209.14988, 2022.

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv.org*, 2103.00020, 2021.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proc. of the International Conf. on Machine learning (ICML)*, 2021.

[31] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[32] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems (NIPS)*, 2022.

[33] B. Schölkopf*, F. Locatello*, S. Bauer, R. Nan Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. Towards causal representation learning. *Proceedings of the IEEE*, 2021.

[34] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models. In *Advances in Neural Information Processing Systems (NIPS)*, 2022.

[35] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *Advances in Neural Information Processing Systems (NIPS)*, 2021.

[36] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3D generation. *arXiv.org*, 2308.16512, 2023.

[37] Christina Tsalicoglou, Fabian Manhardt, Alessio Tonioni, Michael Niemeyer, and Federico Tombari. Textmesh: Generation of realistic 3D meshes from text prompts, 2023.

[38] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3D generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[39] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, 2023.

[40] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[41] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. *arXiv.org*, 2305.16213, 2023.

[42] Jiamin Xu, Zihan Zhu, Hujun Bao, and Weiwei Xu. Hybrid mesh-neural representation for 3D transparent object reconstruction. *arXiv.org*, 2203.12613, 2022.

[43] Jiale Xu, Xintao Wang, Weihao Cheng, Yan-Pei Cao, Ying Shan, Xiaohu Qie, and Shenghua Gao. Dream3d: Zero-shot text-to-3d synthesis using 3d shape prior and text-to-image diffusion models. In *CVPR*, 2023.

[44] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[45] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[46] Guangyao Zhai, Evin Pınar Örnek, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. Commonscenes: Generating commonsense 3d indoor scenes with scene graphs. In *Advances in Neural Information Processing Systems (NIPS)*, 2023.

[47] Hao Zhang, Yao Feng, Peter Kulits, Yandong Wen, Justus Thies, and Michael J Black. Text-guided generation and editing of compositional 3D avatars. *arXiv.org*, 2309.07125, 2023.