Learning to Create 3D Content: Geometry, Appearance, and Physics

Kangle Deng

CMU-RI-TR-25-85

Aug, 2025

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Deva Ramanan, *co-chair*
Jun-Yan Zhu, *co-chair*
Shubham Tulsiani
Maneesh Agrawala, *Stanford University*
Noah Snavely, *Cornell University*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Abstract

With the popularity of Virtual Reality (VR), Augmented Reality (AR), and other 3D applications, developing methods that let everyday users capture and create their own 3D content has become increasingly essential. Current 3D creation pipelines, however, often require tedious manual effort or specialized capture setups. Furthermore, resulting assets frequently suffer from baked-in lighting, inconsistent representations, and a lack of physical plausibility, limiting their use in downstream applications.

This dissertation addresses these challenges by developing methods that leverage data-driven priors to significantly lower the barrier for 3D content creation. By utilizing information from other modalities, large datasets, and pre-trained generative models, the work presented here reduces the burden on user input to casually captured photos, simple sketches, and text prompts.

We first show how depth priors can enable users to digitalize 3D scenes without dense data capture, and discuss how to enable interactive 3D editing and generation through 2D user inputs such as sketches. We then propose an end-to-end text-to-3D generation pipeline that generates both the geometry and texture of 3D assets. For geometry generation, we propose an octree-based adaptive tokenization scheme that allocates representational capacity based on shape complexity, enabling higher-fidelity and more efficient reconstruction and generation of 3D shapes. Moreover, we address appearance modeling by utilizing data and diffusion model priors to generate relightable textures on meshes using text input, ensuring that generated 3D objects are functional in downstream production workflows. Finally, to ground digital designs in reality, we introduce BrickGPT, which incorporates manufacturing and physics constraints to generate physically stable and buildable toy brick structures from text prompts.

Collectively, these contributions bridge the gap between high-level user intent and the creation of editable, functional, and physically realizable 3D content by addressing the core challenges in geometry representation, appearance modeling, and physics-aware generation.

Acknowledgments

I would like to express my deepest gratitude to all those who have supported me throughout my PhD journey. This thesis would not have been possible without their invaluable contributions, guidance, and encouragement.

First and foremost, I extend my sincere appreciation to my advisors, Deva and Jun-Yan. I still remember the summer of 2019 when I was doing a research internship in Deva's lab as an undergraduate student. From that summer, I learned a great deal from Deva, but most importantly, I learned to enjoy doing research. I will always recall the genuine happiness we shared when discussing new insights. Deva, thank you for showing me that research can be so fascinating!

Even before I knew anything about research, I had heard of Jun-Yan's name through his well-known work. He has always been a role model as a brilliant junior faculty with both intellect and diligence. I'm grateful that he taught me to keep polishing every detail of my work. Beyond being an insightful advisor, he is also a close friend in life. He remembers everyone's birthday and orders cake for our lab. In addition to his thoughtful comments on research, his curated list of restaurants in Pittsburgh is another treasure to me.

I feel extremely fortunate to have worked with these two great advisors throughout my PhD.

I am grateful to my committee members, Shubham, Noah, and Maneesh, for their invaluable insights and challenging questions that strengthened this work. I also thank the other mentors I have had the great fortune of working with: Aayush Bansal, Tinghui Zhou, and Kiran Bhat, among others. They have guided me at various stages of my academic journey.

I'm also thankful to my wonderful collaborators: Tianyi Fei, Andrew Liu, Gengshan Yang, Andrew Song, Ruihan Gao, Ruixuan Liu, Yehonathan Litman, Timothy Omernick, Alexander Weiss, Yiheng Zhu, Hsueh-Ti Derek Liu, Xiaoxia Sun, Chong Shang, and Ava Pun, among others. I have learned tremendously from working with each of them.

My heartfelt thanks go to my colleagues and friends at Carnegie Mellon University for making Pittsburgh and Smith Hall feel like home: Sheng-

Yu Wang, Nupur Kumari, Gaurav Parmar, Ruihan Gao, Muyang Li, George Cazenavetteorge, Andrew Song, Beijia Lu, Maxwell Jones, Ava Pun, Sean Liu, Mia Tang, Bingliang Zhang, Daohan Lu, Gengshan Yang, Jeff Tan, Zhiqiu Lin, Tarasha Khurana, Martin Li, Neehar Peri, Jonathon Luiten, Haithem Turki, Aayush Bansal, Yufei Ye, Donglai Xiang among many others. Their friendship and support have made this journey truly memorable.

I extend my thanks to my undergraduate advisors and mentors, Xin Huang, Yuxin Peng, and Jiaying Liu, who provided opportunities to work on exciting projects and first stimulated my interest in research.

I would also like to acknowledge the financial support provided by Microsoft Research PhD Fellowship, without which this research would not have been feasible.

Finally, and most importantly, I am eternally grateful to my family – my parents, my grandparents, and my wife, Yue – for their endless love, patience, understanding, and unwavering belief in me. Their encouragement has always been my constant source of strength and inspiration.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

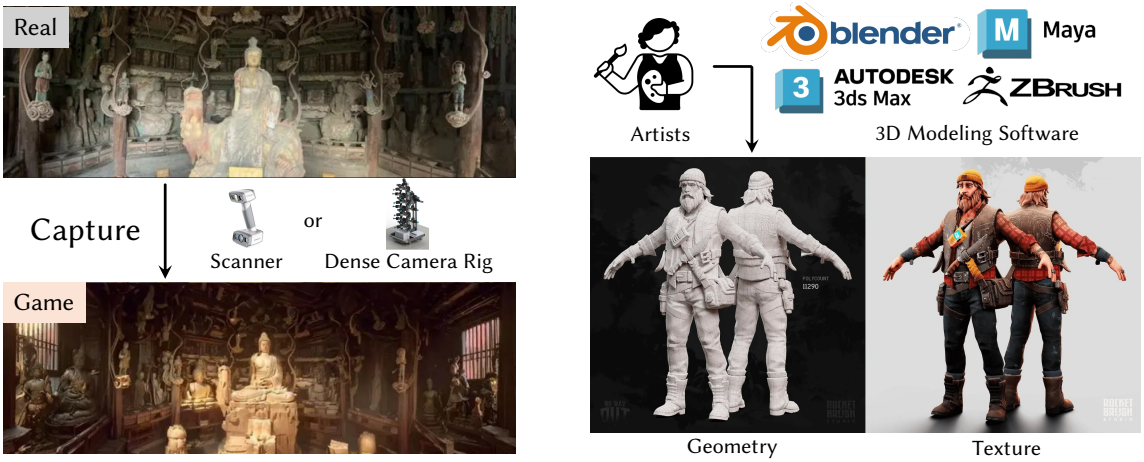
Introduction

1.1 Background

The ability to capture, create, and interact with 3D content is fundamental to the future of digital experiences. From the immersive environments in modern video games and cinematic productions to the expansive virtual worlds of the metaverse [155], the demand for high-fidelity 3D content has never been greater. These technologies promise to benefit fields ranging from entertainment to engineering and robotics, enabling more realistic simulations and creative expressions.

However, a significant bottleneck exists between the demand for 3D content and the ability of most people to produce it. Traditionally, creating high-quality 3D assets has been the domain of specialists, relying on two primary paradigms: ***digitizing the real world*** and ***handcrafting from imagination***.

One important method of creating 3D content is by digitizing the real world [85], as shown in Figure 1.1(a). For example, many video games have their environments created by scanning real-world scenes, which brings a high level of realism. This process, known as photogrammetry [6, 82, 102, 149, 167, 199, 263] or 3D scanning [35, 76, 187, 219], can produce assets with unparalleled fidelity. Another way of 3D content creation is handcrafting them as illustrated in Figure 1.1(b). Artists use professional 3D modeling software, e.g., Blender, Maya, and 3DSMax, to design and build virtual worlds, shaping every detail from scratch. This approach allows for boundless creativity, enabling the creation of fantastic objects and scenes



(a) 3D Digitization requires dense views or special setup.

(b) 3D Modeling requires significant expertise.

Figure 1.1: **Two ways of creating 3D content.** (a) **3D Digitization** captures real-world environments through photogrammetry or 3D scanning, requiring dense camera views or specialized equipment setups to achieve high-fidelity results. (b) **3D Modeling** involves artists using professional software to handcraft virtual content from scratch, including both geometry and texture creation. Both approaches require specialized expertise and remain inaccessible to everyday users.

that do not exist in reality.

However, both of these methods are mainly used by professionals and remain out of reach for everyday users. 3D scanning and photogrammetry often require densely captured views [6, 82, 102, 149, 167, 263] or even specialized equipment setups [76, 219]. Meanwhile, 3D modeling demands significant expertise, and mastering the software and techniques can be time-consuming and complex.

1.2 Challenges

Recent advances in Neural Radiance Fields (NeRF) [149] and Gaussian Splatting (GS) [90] have made it possible to digitize 3D scenes using only 2D images. Simultaneously, the emergence of generative AI has revolutionized content creation across multiple domains, from text generation with large language models [2, 44] to image synthesis with GANs [87] and diffusion models [192]. However, extending these successes to usable 3D content creation, whether through reconstructing existing

scenes or generating novel assets, presents unique and fundamental challenges.

Reconstruction Challenges. The most fundamental challenge in 3D reconstruction stems from the inherent ambiguity of inferring 3D structure from 2D observations. Multiple 3D configurations can produce identical 2D projections, creating fundamental ambiguities that existing methods, e.g., NeRF [149] and GS [90], resolve by requiring hundreds of densely captured images from diverse viewpoints. This dense view requirement severely restricts practical applications, as capturing such extensive datasets is often impractical or impossible in real-world scenarios where users want to quickly digitize objects or scenes with minimal effort. The problem is compounded by the need for careful camera positioning, controlled lighting conditions, and often specialized equipment to achieve the coverage and quality necessary for high-fidelity reconstruction.

Generation Challenges. While 2D generation often focuses on visual appeal from a single viewpoint, generated 3D content must appear coherent across all possible views. This multi-view consistency requirement dramatically increases the complexity of the generation task, as models must reason about the underlying 3D content that looks plausible from every perspective rather than optimizing for a single view. The challenge is exacerbated by severe data scarcity in the 3D domain. While 2D image datasets like LAION [201] contain billions of examples, high-quality 3D datasets are orders of magnitude smaller and significantly more expensive to create, requiring specialized equipment, e.g., LiDAR scanners, or manually creating 3D assets. This data scarcity makes it challenging to train robust generative models that can handle the full diversity of real-world 3D content. Additionally, 3D content creation involves multiple interconnected modalities, geometry, appearance, and physics, each with its own representation challenges and constraints. Unlike 2D generation where visual quality is often the primary objective, 3D content frequently needs to satisfy additional requirements depending on its intended use, e.g., efficient geometry representations that adapt to different levels of detail, correct material properties for realistic lighting and interaction, and physical plausibility for downstream simulation or manufacturing purposes.

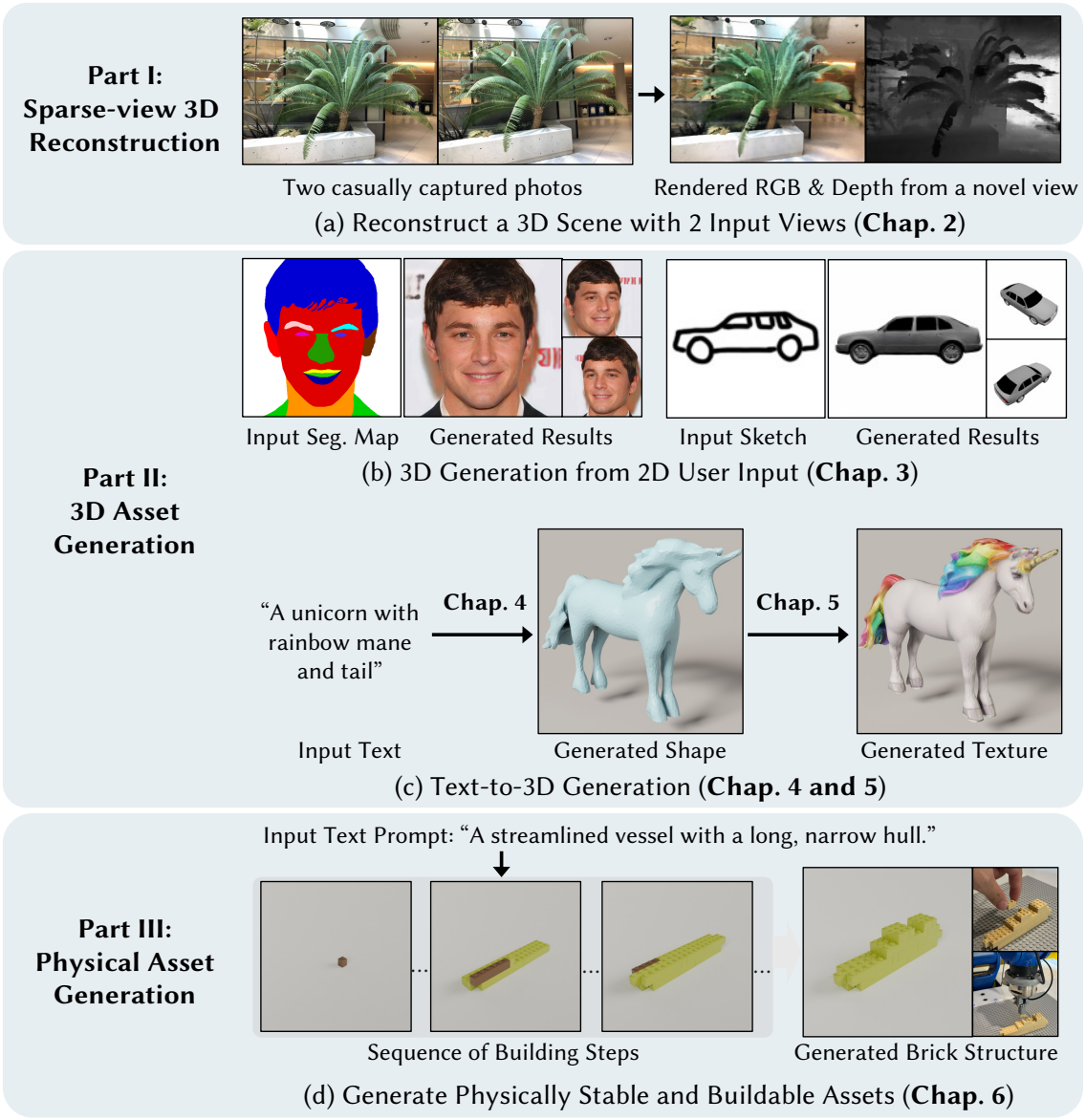


Figure 1.2: **Dissertation Overview.** This dissertation is organized in three parts: (a) **Sparse-view 3D Reconstruction** (Part I) enables high-quality 3D scene reconstruction from as few as two casually captured photos. (b)(c) **3D Asset Generation** (Part II) creates 3D content from intuitive inputs: generating editable 3D objects from 2D segmentation maps or sketches for category-specific generation (Chapter 3), and diverse 3D asset creation from text descriptions (Chapters 4 and 5). (d) **Physical Asset Generation** (Part III) extends beyond virtual assets to generate physically stable and buildable brick structures from text prompts, providing step-by-step construction sequences for real-world implementation.

1.3 Dissertation Overview

To address the challenges outlined above, this dissertation proposes to integrate classical 3D understanding with modern generative models across the core aspects of 3D content: *geometry*, *appearance*, and *physics*. Rather than relying purely on data-driven learning, our approach leverages well-established geometric principles, structural knowledge, and physical constraints developed over decades of 3D computer vision and graphics research. By grounding generative models in classical 3D principles, e.g., multi-view geometry, octree hierarchy, material representations, and structural stability, we can create more robust, efficient, and practically useful 3D generation systems.

Specifically, this dissertation is structured in three parts. We begin by making real-world capture easier, then move to simplifying creative generation, and conclude by bridging the gap between digital designs and physically realizable objects.

Part I: Sparse-View 3D Reconstruction. It is vital to develop a generic and easy-to-use method to reconstruct objects and scenes in real life. Neural Radiance Fields (NeRF) [149] make this possible using only 2D images. However, plain NeRF and many of its follow-up works [263] require dense views from hundreds of images, which restricts its usage. Part I aims to lower the bar of using 3D reconstruction in our daily life by relieving the burden of taking dense view data.

As shown in Figure 1.2(a), Chapter 2 introduces Depth-Supervised NeRF (DS-NeRF) [39], a method that utilizes geometric priors using "free" depth information obtained from standard Structure-from-Motion (SFM) pipelines [199]. By supervising the geometry learning process, DS-NeRF reduces overfitting, accelerates training, and enables high-quality view synthesis from as few as two images, making high-fidelity capture accessible from casual photos. Building on this foundation, our follow-up work Total-recon [212], while not included in the thesis, shows that depth supervision can also help improve 4D reconstruction from monocular videos.

Part II: 3D Asset Generation. In addition to capturing 3D scenes from the real world, we also want to lower the skill barrier for creating novel 3D content. Part II explores methods that generate rich 3D assets from intuitive, low-effort inputs such as 2D sketch maps and text prompts.

Chapter 3 presents a method for 3D-aware conditional image synthesis [40], which can generate editable 3D objects of specific categories, e.g., cars, from a single 2D semantic input like a segmentation or sketch map, as shown in Figure 1.2(b). Notably, by interactive editing of the semantic maps projected onto user-specified viewpoints, our system can be used as a tool for 3D editing of the generated content.

Expanding beyond visual inputs and category-specific asset generation, the following two chapters explore natural language as an interface for generic 3D asset creation. Similar to how artists create 3D assets, we divide the problem into a two-stage process, geometry generation (Chapter 4) and texture generation (Chapter 5), as shown in Figure 1.2(c).

Specifically, chapter 4 introduces a more efficient way to represent and generate 3D shapes with Octree-based Adaptive Tokenization (OAT) [42]. This method dynamically allocates representational capacity based on geometric complexity. Building on this, OctreeGPT can generate diverse, high-quality 3D shapes from text descriptions, advancing on prior text-to-3D methods [106, 179, 223].

Chapter 5 addresses the crucial challenge of appearance modeling by generating high-quality relightable texturing. The proposed method, FlashTex [41], textures an input 3D mesh given a user-provided text prompt, improving upon prior texturing methods [23, 24, 189]. Notably, our generated texture can be relit properly in different lighting environments, which makes our results widely applicable in downstream tasks.

Part III: Physical Asset Generation. In addition to virtual assets, we also explore a novel direction by considering the physical realizability of generated designs [133]. While most generative models produce objects that are only visually plausible, this final part of the thesis explores how to create designs that are structurally sound and buildable in the real world.

Chapter 6 introduces BrickGPT [181], a system that generates physically stable and buildable brick structures directly from text prompts. By repurposing a large language model [44] for "next-brick prediction" and incorporating physics-aware checks for stability [121], BrickGPT produces complex and creative designs that are not only imaginative but also physically stable and buildable, as shown in Figure 1.2(d).

1.4 Other Research

In addition to the thesis work, I also contributed to other relevant research projects in 3D content creation, including monocular video reconstruction [212], texture generation with tactile input [48], stability analysis [119], 3D reconstruction with materials [113], and scene generation [191]:

1. Chonghyuk Song, Gengshan Yang, **Kangle Deng**, Jun-Yan Zhu and Deva Ramanan. *Total-Recon: Deformable Scene Reconstruction for Embodied View Synthesis.* ICCV 2023. [212]
2. Ruihan Gao, **Kangle Deng**, Gengshan Yang, Wenzhen Yuan, Jun-Yan Zhu. *Tactile DreamFusion: Exploiting Tactile Sensing for 3D Generation.* NeurIPS 2024. [48]
3. Ruixuan Liu, **Kangle Deng**, Ziwei Wang, Changliu Liu. *StableLego: Stability Analysis of Block Stacking Assembly*. RA-L 2024. [119]
4. Yehonathan Litman, Or Patashnik, **Kangle Deng**, Aviral Agrawal, Rushikesh Zawar, Fernando De la Torre, Shubham Tulsiani. *MaterialFusion: Enhancing Inverse Rendering with Material Diffusion Priors*. 3DV 2025. [113]
5. Roblox Foundation AI Team: **Kangle Deng** (core contributor). *Cube: A Roblox View of 3D Intelligence.* Technical Report 2025. [191]

Part I

Sparse-view 3D Reconstruction

Chapter 2

Depth-supervised NeRF: Fewer Views and Faster Training for Free

We begin our exploration of 3D content creation by addressing the challenge of sparse-view 3D reconstruction. As established in the challenges section, traditional 3D reconstruction methods suffer from inherent ambiguity when inferring 3D structure from limited 2D observations, typically requiring hundreds of densely captured images to achieve high-quality results.

The key insight driving this chapter is that we can resolve reconstruction ambiguity not by requiring more data, but by incorporating complementary geometric priors that are readily available from standard reconstruction pipelines.

The following chapter demonstrates how this integration of classical multi-view geometry with modern neural rendering can dramatically reduce view requirements while maintaining reconstruction quality, making 3D scene capture accessible from casual photographs.

2.1 Introduction

Neural rendering with implicit representations has become a widely-used technique for solving many vision and graphics tasks ranging from view synthesis [32, 149, 210], to re-lighting [134, 143], to pose and shape estimation [169, 195, 271], to 3D-aware image synthesis and editing [17, 124, 202], to modeling dynamic

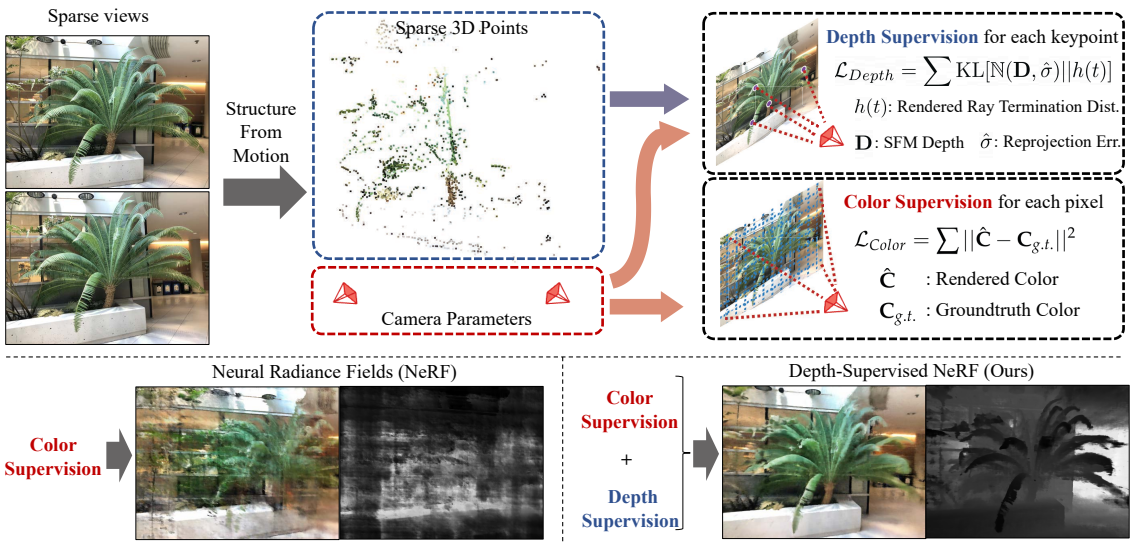


Figure 2.1: Training NeRFs can be difficult when given insufficient input images. We utilize additional supervision from depth recovered from 3D point clouds estimated from running structure-from-motion and impose a loss to ensure the rendered ray's termination distribution respects the surface priors given by the each keypoint. Because our supervision is complementary to NeRF, it can be combined with any such approach to reduce overfitting and speed up training.

scenes [108, 172, 180]. The seminal work of Neural Radiance Fields (NeRF) [149] demonstrated impressive view synthesis results by using implicit functions to encode volumetric density and color observations.

In spite of this, NeRF has several limitations. Reconstructing both the scene appearance and geometry can be ill-posed given a small number of input views. Figure 2.2 shows that NeRF can learn wildly inaccurate scene geometries that still accurately render train-views. However, such models produce poor renderings of novel test-views, essentially overfitting to the train set. Furthermore, even given a large number of input views, NeRF can still be time-consuming to train; it often takes between ten hours to several days to model a single scene at moderate resolutions on a single GPU. The training is slow due to both the expensive ray-casting operations and lengthy optimization process.

In this work, we explore depth as an additional, cheap source of supervision to guide the geometry learned by NeRF. Typical NeRF pipelines require images and camera poses, where the latter are estimated from structure-from-motion (SFM)

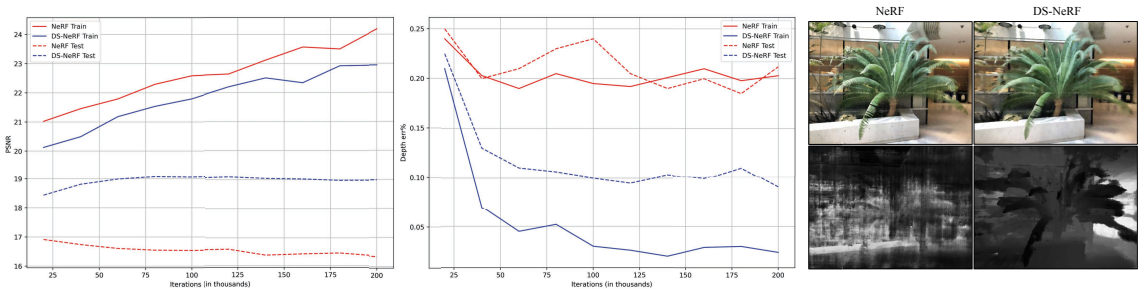


Figure 2.2: **Few view NeRF.** NeRF is susceptible to overfitting when given few training views. As seen by the PSNR gap between train and test renders (left), NeRF has overfit and fails at synthesizing novel views. Further, the depth map (right) and depth error (middle) for NeRF suggest that its density function has failed to extract the surface geometry and can only reconstruct the training views' colors. Our depth-supervised NeRF model is able to render plausible geometry with consistently lower depth errors.

solvers such as COLMAP [199]. In addition to returning cameras, COLMAP also outputs sparse 3D point clouds as well as their reprojection errors. We impose a loss to encourage the distribution of a ray's termination to match the 3D keypoint, incorporating reprojection error as an uncertainty measure. This is a significantly stronger signal than reconstructing only RGB. Without depth supervision, NeRF is implicitly solving a 3D correspondence problem between multiple views. However, the sparse version of this exact problem has already been solved by SFM, whose solution is given by the sparse 3D keypoints. Therefore depth supervision improves NeRF by (softly) anchoring its search over implicit correspondences with sparse explicit ones.

Our experiments show that this simple idea translates to massive improvements in training NeRFs and its variations, regarding both the training speed and the amount of training data needed. We observe that depth-supervised NeRF can accelerate model training by 2-3x while producing results with the same quality. For sparse view settings, experiments show that our method synthesizes better results compared to the original NeRF and recent sparse-views NeRF models [220, 273] on both NeRF Real [149] and Redwood-3dscan [33] We also show that our depth supervision loss works well with depth derived from other sources such as a depth camera. Our code and more results are available at https://www.cs.cmu.edu/~dsnerf/.

2.2 Related Work

NeRF from few views. NeRF [149] was originally shown to work on a large number of images with the LLFF NeRF Real dataset [148] consisting of nearly 50 images per scene. This is because fitting the NeRF volume often requires a large number of views to avoid arriving at degenerate representations. Recent works have sought to decrease the data-hungriness of NeRF in a variety of different ways. PixelNeRF [273] and metaNeRF [220] use data-driven priors recovered from a domain of training scenes to fill in missing information from test scenes. Such an approach works well when given sufficient training scenes and limited gap between the training and test distribution, but such assumptions are not particularly flexible. Another approach is to leverage priors recovered from a different task like semantic consistency [78] or depth prediction [254].

Similar to our insight that the primary difficulty in fitting few-view NeRF is correctly modeling 3D geometry, MVSNeRF [21] combines both 3D knowledge with scene priors by constructing a plane sweep volume before using a pretrained network with generalizable priors to render scenes. One appeal of an approach that utilizes 3D information is the lack of assumptions it makes on the problem statement. Unlike the aforementioned approaches which depend on the availability of training data or the applicability of prior assumptions, our approach only requires the existence of 3D keypoints. This gives depth supervision the flexibility to be used not only as a standalone method, but one that can be freely incorporated into existing NeRF methods easily.

Faster NeRF. Another drawback of NeRF is the lengthy optimization time required to fit the volumetric representation. Indeed Mildenhall *et al*. [149] trained a single scene's NeRF model for twelve hours of GPU compute. Many works [186, 272] have found that the limiting factor is not learning the radiance itself, but rather oversampling the empty space during training. Indeed this is a similar intuition to the fact that the majority of the volume is actually empty, but NeRF's initialization is a median uniform density. Our insight is to apply a supervisory signal directly to the NeRF density to increase the convergence of the geometry and to encourage NeRF's density function to mimic the behavior of real-world surface geometries.

Depth and NeRF. Several prior works have explored ways to leverage depth information for view synthesis [205, 234] and NeRF training [108, 114, 158, 172, 254]. For instance, 3D keypoints have been demonstrated to be helpful when extending NeRFs with relaxed assumptions like deformable surfaces [172] or dynamic scene flows [108]. Other works like DONeRF [158] proposed training a depth oracle to improve rendering speed by directly smartly sampling the surface of a NeRF density function. Similar to DONeRF, NerfingMVS [254] shows how a monocular depth network can be used to induce depth priors to do smarter sampling during training and inference.

Our work attempts to improve NeRF-based methods by directly supervising the NeRF density function. As depth becomes a more accessible source of data, being able to apply depth supervision becomes increasingly more powerful. For example, recent works have demonstrated how depth extracted from sensors like time-of-flight cameras [4] or RGB-D Kinect sensor [5] can be applied to fit implicit functions. Building upon their insights, we provide a probabilistic formulation of the depth supervision, and show this results in meaningful improvements to NeRF and its variants.

2.3 Depth-Supervised Ray Termination

We now present our proposed depth-supervised loss for training NeRFs. We first revisit volumetric rendering and then analyze the termination distribution for rays. We conclude with our depth-supervised distribution loss.

2.3.1 Volumetric rendering revisited

A Neural Radiance Field takes a set of posed images and encodes a scene as a volume density and emitted radiance. More specifically, for a given 3D point $\mathbf{x} \in \mathbb{R}^3$ and a particular viewing direction $\mathbf{d} \in \mathbb{R}^3$, NeRF learns an implicit function f that estimates the differential density σ and RGB color $\mathbf{c}$ like so: $f(\mathbf{x}, \mathbf{d}) = (\sigma, \mathbf{c})$.

To render a 2D image given a pose $\mathbf{P}$, we cast rays $\mathbf{r}$ originating from the $\mathbf{P}$'s center of projection $\mathbf{o}$ in direction $\mathbf{d}$ derived from its intrinsics. We integrate the implicit radiance field along this ray to compute the incoming radiance from any

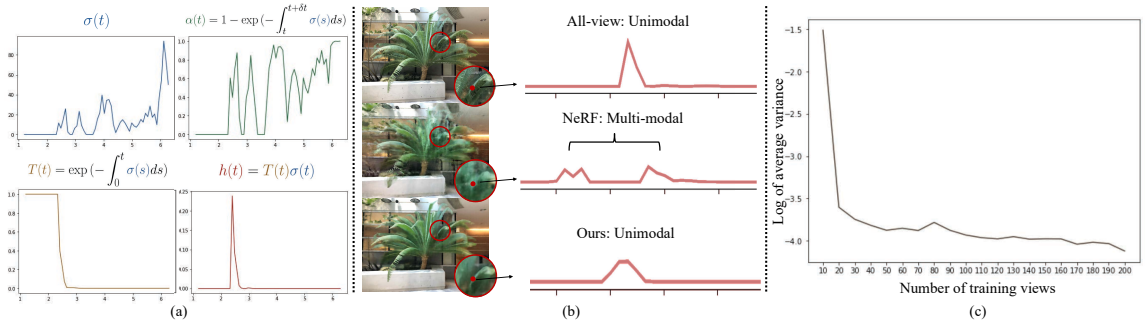


Figure 2.3: **Ray Termination Distribution.** (a) We plot various NeRF components over the distance traveled by the ray. Even if a ray traverses through multiple objects (as indicated by the multiple peaks of density $\sigma(t)$), we find that the ray termination distribution $h(t)$ is still unimodal. We find that NeRF models trained with sufficient supervision tend to have peaky, unimodal ray termination distributions as seen by the decreasing variance with more views in (c). We posit that the ideal ray termination distribution approaches a δ impulse function.

object that lies along $\mathbf{d}$:

$$\hat{\mathbf{C}} = \int_0^\infty T(t)\sigma(t)\mathbf{c}(t)dt, \tag{2.1}$$

where t parameterizes the aforementioned ray as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ and $T(t) = \exp(-\int_0^t \sigma(s)ds)$ checks for occlusions by integrating the differential density between 0 to t. Because the density and radiance are the outputs of neural networks, NeRF methods approximate this integral using a sampling-based Riemann sum instead. The final NeRF rendering loss is given by a reconstruction loss over colors returned from rendering the set of rays $\mathcal{R}(\mathbf{P})$ produced by a particular camera parameter $\mathbf{P}$.

$$\mathcal{L}_{\text{Color}} = \mathbb{E}_{\mathbf{r}\in\mathcal{R}(\mathbf{P})} \left\| \hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2. \tag{2.2}$$

Ray distribution. Let us write $h(t) = T(t)\sigma(t)$. In the appendix, we show that it is a continuous probability distribution over ray distance t that describes the likelihood of a ray terminating at t. Due to practical constraints, NeRFs assume that the scene lies between a near and far bound (t_n, t_f). To ensure $h(t)$ sums to one, NeRF implementations often treat t_f as an opaque wall. With this definition,

the rendered color can be written as an expectation:

$$\hat{\mathbf{C}} = \int_0^\infty h(t)\mathbf{c}(t)dt = \mathbb{E}_{h(t)}[\mathbf{c}(t)].$$

Idealized distribution. The distribution $h(t)$ describes the weighed contribution of sampled radiances along a ray to the final rendered value. Most scenes consist of empty spaces and opaque surfaces that restrict the weighted contribution to stem from the closest surface. This implies that the ideal ray distribution of image point with a closest-surface depth of $\mathbf{D}$ should be $\delta(t - \mathbf{D})$. Figure 2.3(c) shows that the empirical variance of NeRF termination distributions decreases with more training views, suggesting that high quality NeRFs (trained with many views) tend to have ray distributions that approach the δ-function. This insight motivates our depth-supervised ray termination loss.

2.3.2 Deriving depth-supervision

Recall that most NeRF pipelines require images with associated camera matrices $(\mathbf{P}_1, \mathbf{P}_2, \ldots)$, often estimated with SFM packages such as COLMAP [199]. Importantly, SFM makes use of bundle adjustment, which also returns 3D keypoints $\{\mathbf{X} : \mathbf{x}_1, \mathbf{x}_2, \ldots \in \mathbb{R}^3\}$ and visibility flags for which keypoints are seen from camera j: $\mathbf{X}_j \subset \mathbf{X}$. Given image I_j and its camera $\mathbf{P}_j$, we estimate the depth of visible keypoints $\mathbf{x}_i \in \mathbf{X}_j$ by simply projecting $\mathbf{x}_i$ with $\mathbf{P}_j$, taking the re-projected z value as the keypoint's depth $\mathbf{D}_{ij}$.

Depth uncertainty. Unsurprisingly $\mathbf{D}_{ij}$ are inherently noisy estimates due to spurious correspondences, noisy camera parameters, or poor COLMAP optimization. The reliability of a particular keypoint $\mathbf{x}_i$ can be measured using the average reprojection error $\hat{\sigma}_i$ across views over which the keypoint was detected. Specifically, we model the location of the first surface encountered by a ray as a random variable $\mathbb{D}_{ij}$ that is normally distributed around the COLMAP-estimated depth $\mathbf{D}_{ij}$ with variance $\hat{\sigma}_i$: $\mathbb{D}_{ij} \sim \mathbb{N}(\mathbf{D}_{ij}, \hat{\sigma}_i)$. Combining the intuition regarding behavior of ideal termination distribution, our objective is to minimize the KL divergence between the rendered ray distribution $h_{ij}(t)$ of $\mathbf{x}_i$'s image coordinates and the noisy depth

distribution:

$$\min_{h_{ij}} \mathbb{E}_{\mathbb{D}_{ij}} \mathrm{KL}[\delta(t - \mathbf{D}_{ij}) || h_{ij}(t)] \tag{2.3}$$

$$= \arg\min_{h} \mathbb{E}_{\mathbf{D}_{g.t.}} \int p(t; \mathbf{D}_{g.t.}) \log \frac{p(t; \mathbf{D}_{g.t.})}{h(t)} dt \tag{2.4}$$

$$\Leftrightarrow \max_{h_{ij}} \mathbb{E}_{\mathbb{D}_{ij}} \int \delta(t - \mathbf{D}_{ij}) \log h_{ij}(t) dt \tag{2.5}$$

$$\Leftrightarrow \max_{h_{ij}} \mathbb{E}_{\mathbb{D}_{ij}} \log h_{ij}(\mathbf{D}_{ij}) \tag{2.6}$$

$$\Leftrightarrow \max_{h_{ij}} \int \log h_{ij}(t) \exp -\frac{(t - \mathbf{D}_{ij})^2}{2\sigma_i^2} dt. \tag{2.7}$$

$$\Leftrightarrow \min_{h_{ij}} \mathrm{KL}[\mathbb{N}(\mathbf{D}_{ij}, \sigma_i) || h_{ij}(t)]. \tag{2.8}$$

$$\mathbb{E}_{\mathbb{D}_{ij}} \mathrm{KL}[\delta(t - \mathbb{D}_{ij}) || h_{ij}(t)] = \mathrm{KL}[\mathbb{N}(\mathbf{D}_{ij}, \hat{\sigma}_i) || h_{ij}(t)] + const.$$

Ray distribution loss. The above equivalence (see our appendix for proof) allows the termination distribution $h(t)$ to be trained with probabilisitic COLMAP depth supervision:

$$\begin{aligned} \mathcal{L}_{Depth} &= -\mathbb{E}_{x_i \in X_j} \int \log h(t) \exp\left(-\frac{(t - \mathbf{D}_{ij})^2}{2\hat{\sigma}_i^2}\right) dt \\ &\approx -\mathbb{E}_{x_i \in X_j} \sum_k \log h_k \exp\left(-\frac{(t_k - \mathbf{D}_{ij})^2}{2\hat{\sigma}_i^2}\right) \Delta t_k. \end{aligned}$$

Our overall training loss for NeRF is $\mathcal{L} = \mathcal{L}_{\text{Color}} + \lambda_D \mathcal{L}_{\text{Depth}}$ where λ_D is a hyperparameter balancing color and depth supervision.

2.4 Experiments

We first evaluate the input data efficiency on view synthesis over several datasets in Section 2.4.3. For relevant NeRF-related methods, we also evaluate the error of rendered depth maps in Section 2.4.4. Finally, we analyze training speed improvements in Section 2.4.5.

NeRF Real [148]	PSNR↑ 2-view	PSNR↑ 5-view	PSNR↑ 10-view	SSIM↑ 2-view	SSIM↑ 5-view	SSIM↑ 10-view	LPIPS↓ 2-view	LPIPS↓ 5-view	LPIPS↓ 10-view
LLFF	14.3	17.6	22.3	0.48	0.49	0.53	0.55	0.51	0.53
NeRF	13.5	18.2	22.5	0.39	0.57	0.67	0.56	0.50	0.52
metaNeRF-DTU	13.1	13.8	14.3	0.43	0.45	0.46	0.89	0.88	0.87
pixelNeRF-DTU	9.6	9.5	9.7	0.39	0.40	0.40	0.82	0.87	0.81
finetuned	18.2	22.0	24.1	0.56	0.59	0.63	0.53	0.53	0.41
finetuned w/ DS	18.9	22.1	24.4	0.54	0.61	0.66	0.55	0.47	0.42
IBRNet	14.4	21.8	24.3	0.50	0.51	0.54	0.53	0.54	0.51
finetuned w/ DS	19.3	22.3	24.5	0.63	0.66	0.68	**0.39**	0.36	0.38
MVSNeRF	-	17.2	17.2	-	0.61	0.60	-	0.37	0.36
fintuned	-	21.8	22.9	-	**0.70**	0.74	-	**0.27**	**0.23**
fintuned w/ DS	-	22.0	22.9	-	**0.70**	**0.75**	-	0.27	0.25
DS-NeRF									
MSE	19.5	22.2	24.7	0.65	0.69	0.71	0.43	0.40	0.37
KL divergence	**20.2**	**22.6**	**24.9**	**0.67**	0.69	0.72	**0.39**	0.35	0.34

Table 2.1: **View Synthesis on NeRF Real.** We evaluate view synthesis quality for various methods when given 2, 5, 10 views from NeRF Real. We find that metaNeRF-DTU and pixelNeRF-DTU struggle to learn on NeRF Real due to its domain gap to DTU. PixelNeRF, IBRNet and MVSNeRF can benefit from incorporating the depth supervision loss to achieve their best performance. We find that our DS-NeRF outperforms these methods on a variety of metrics, but especially for the few view settings like 2 and 5 views.

DTU [79]	PSNR↑ 3-view	PSNR↑ 6-view	PSNR↑ 9-view	SSIM↑ 3-view	SSIM↑ 6-view	SSIM↑ 9-view	LPIPS↓ 3-view	LPIPS↓ 6-view	LPIPS↓ 9-view
NeRF	9.9	18.6	22.1	0.37	0.72	**0.82**	0.62	0.35	0.26
metaNeRF-DTU	18.2	18.8	20.2	0.60	0.61	0.67	0.40	0.41	0.35
pixelNeRF-DTU	**19.3**	20.4	21.1	**0.70**	0.73	0.76	**0.39**	0.36	0.34
DS-NeRF									
MSE	16.5	20.5	22.2	0.54	0.73	0.77	0.48	0.31	0.26
KL divergence	16.9	**20.6**	**22.3**	0.57	**0.75**	0.81	0.45	**0.29**	**0.24**

Table 2.2: **View Synthesis on DTU.** We evaluate on 3, 6, and 9 views respectively for 15 test scenes from the DTU dataset. pixelNeRF-DTU and metaNeRF-DTU perform well given that the domain overlap between training and testing. This is especially true for the few view setting as the lack of information is supplemented by exploiting dataset priors. In spite of this, DS-NeRF is still competitive on view synthesis for 6 and 9 views.

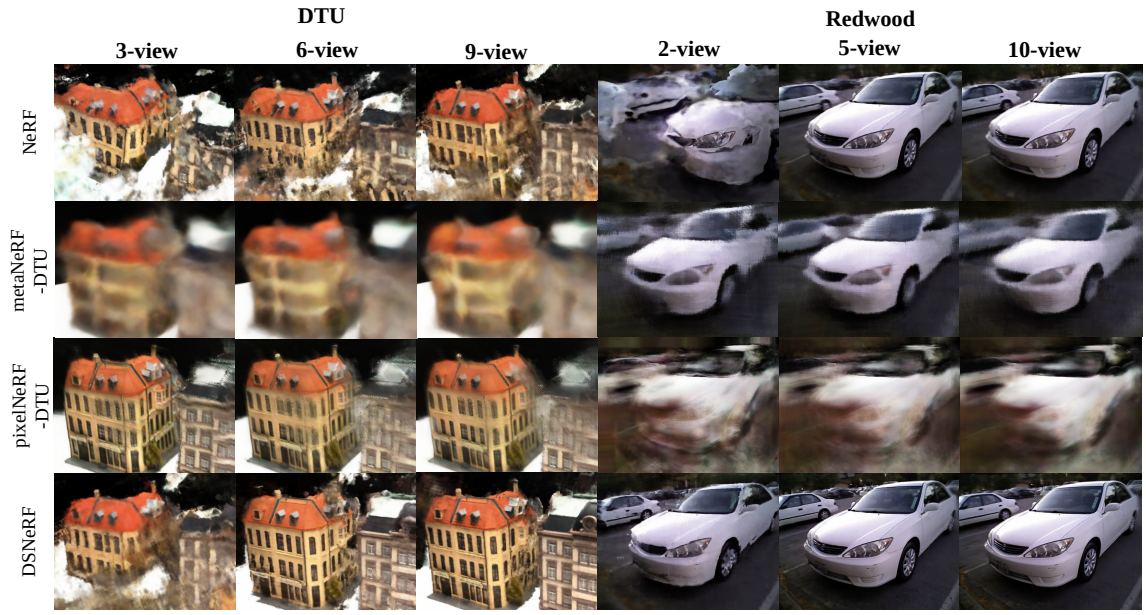


Figure 2.4: **View Synthesis on DTU and Redwood.** PixelNeRF, which is pre-trained on DTU, performs the best when given 3-views, although we find DS-NeRF to be visually competitive when more views are available. On Redwood, DS-NeRF is the only baseline to perform well on the 2-views setting.

2.4.1 Datasets

DTU MVS Dataset (DTU) [79] captures various objects from multiple viewpoints. Following Yu *et al.*'s setup in PixelNeRF [273], we evaluated on the same test scenes and views. For each scene, we used their subsets of size 3, 6, 9 training views. We run COLMAP with the ground truth calibrated camera poses to get keypoints. Images are down-sampled to a resolution of 400×300 for training and evaluation.

NeRF Real-world Data (NeRF Real) [148, 149] contains 8 real world scenes captured from many forward-facing views. We create subsets of training images for each scene of sizes 2, 5, and 10 views. For every subset, we run COLMAP [199] over its training images to estimate cameras and collect sparse keypoints for depth supervision.

Redwood-3dscan (Redwood) [33] contains RGB-D videos of various objects. We select 5 RGB-D sequences and create subsets of 2, 5, and 10 training frames for each object. We run COLMAP to get their camera poses and sparse point clouds.

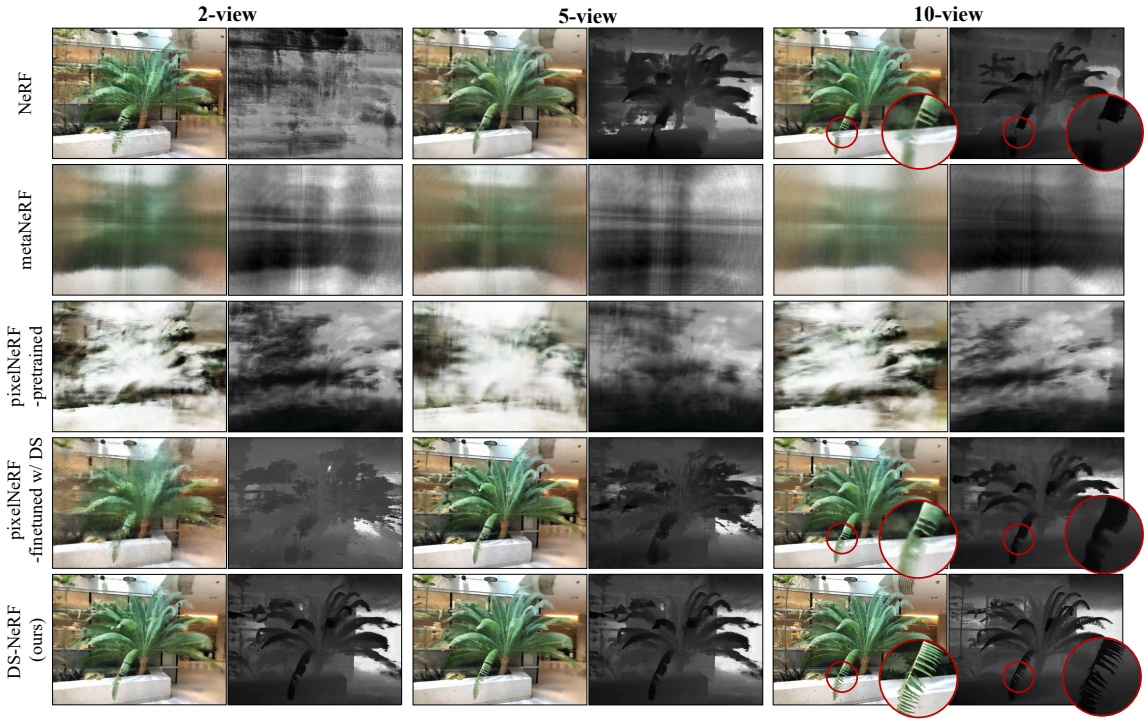


Figure 2.5: **Qualitative Comparison on NeRF Real.** We render novel views and depth for various NeRF models trained on 2, 5, and 10 views. We find that methods trained with DTU struggle on NeRF Real while methods that use depth-supervision are able to render test views with realistic depth maps, even when only 2 views are provided. Please refer to Table 2.1 for quantitative comparisons.

To connect the scale of COLMAP's pose with the scanned depth, we solve a least-squares that best fits detected keypoints to the scanned depth value. Please refer to our appendix for full details.

2.4.2 Comparisons

First we consider Local Lightfield Fusion (*LLFF*) [148], an MPI-based representation that learns from multiple view points. Next we consider a set of NeRF baselines.

PixelNeRF [273] expands upon NeRF by using an encoder to train a general model across multiple scenes. *pixelNeRF-DTU* is evaluated using the released DTU checkpoint. For cases where the train and test domain are different, we finetune using RGB supervision for additional iterations on each test scene to get *pixelNeRF finetuned*.

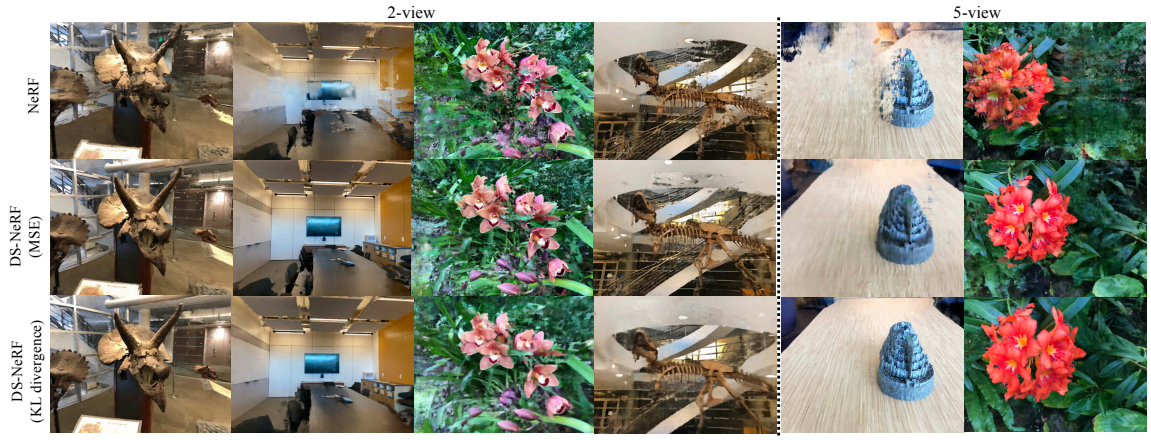


Figure 2.6: **Depth Supervision Ablations.** We render novel views for NeRF and DS-NeRF trained on 2 views and 5 views. NeRF fails to render novel views as evident by the many artifacts. Using MSE between rendered and sparse depth improves results slightly, but with KL Divergence, DS-NeRF is able to render images with the fewest artifacts.

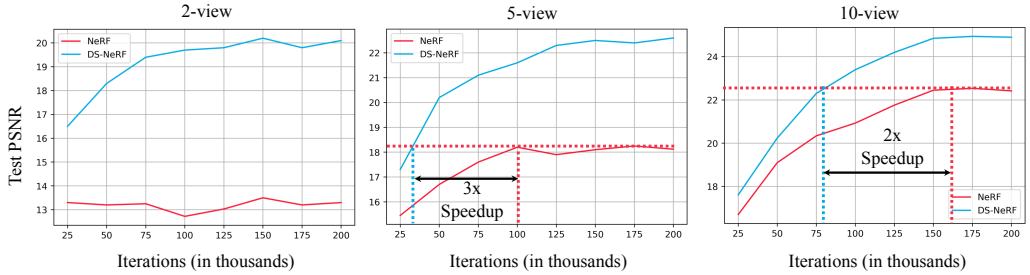


Figure 2.7: **Faster Training.** We train DS-NeRF and NeRF under identical conditions and observe that DS-NeRF is able to reach NeRF's peak PSNR quality in a fraction of the number of iterations across. For 2 views, we find that NeRF is unable to match DS-NeRF's performance.

MetaNeRF [220] finds a better NeRF initialization over a domain of training scenes before running test-time optimization on new scenes. Because DTU is the only dataset large enough for meta-learning, we only consider the *metaNeRF-DTU* baseline which learns an initialization over DTU for 40K meta-iterations and then finetunes for 1000 steps on new scenes. We follow metaNeRF's ShapeNet experiments to demonstrate its susceptibility to differences between training and testing domains.

IBRNet [245] extends NeRF by using a MLP and ray transformer to estimate radiance and volume density.

Redwood-3dscan [33]	PSNR↑ 2-view	5-view	10-view	SSIM↑ 2-view	5-view	10-view	LPIPS↓ 2-view	5-view	10-view
NeRF	10.5	22.4	23.4	0.38	0.75	0.82	0.51	0.45	0.45
metaNeRF-DTU	14.3	14.6	15.1	0.37	0.39	0.40	0.76	0.76	0.75
pixelNeRF-DTU	12.7	12.9	12.8	0.43	0.47	0.50	0.76	0.75	0.70
MVSNeRF-DTU	-	17.1	17.1	-	0.54	0.53	-	0.63	0.63
finetuned	-	22.7	23.1	-	0.78	0.78	-	0.36	0.34
DS-NeRF	18.1	22.9	23.8	0.62	**0.78**	0.81	0.40	**0.34**	0.42
DS-NeRF w/ RGB-D	**20.3**	**23.4**	**23.9**	**0.73**	0.77	**0.84**	**0.36**	0.35	**0.28**

Table 2.3: **View Synthesis on Redwood.** We evaluate view synthesis on 2, 5, and 10 input views on the Redwood dataset. DS-NeRF (with COLMAP [199] inputs) outperforms baselines on various metrics across varying numbers of views. Learning DS-NeRF with the RGB-D reconstruction output [276] further improves performance, highlighting the potential of applying our method alongside other sources of depth.

MVSNeRF [21] initializes a plane sweep volume from 3 views before converting it to a NeRF by a pretrained network. MVSNeRF can be further optimized using RGB supervision.

DS-NeRF (Ours). To illustrate the effectiveness of KL divergence, we include a variant of DS-NeRF with an MSE loss between the SFM-estimated and the rendered depth. Figure 2.6 qualitatively shows that KL divergence penalty produces views with less artifacts on NeRF Real sequences.

DS with existing methods. As our DS loss does not require additional annotation or assumptions, our loss can be inserted into many NeRF-based methods. Here, we also incorporate our loss when finetuning pixelNeRF and IBRNet.

2.4.3 Few-input view synthesis

We start by comparing each method on rendering test views from few inputs. For view synthesis, we report three metrics (PSNR, SSIM [251], and LPIPS [285]) that evaluate the quality of rendered views against a ground truth.

DTU. We show evaluations on DTU in Table 2.2 and qualitative results in Figure 2.4. We find that DS-NeRF renders images from 6 and 9 input views that are competitive with pixelNeRF-DTU, however metaNeRF-DTU and pixelNeRF-DTU are able to outperform DS-NeRF on 3-views. This is not particularly surprising as both

Depth err%↓	NeRF real-world			Redwood-3dscan		
	2-view	5-view	10-view	2-view	5-view	10-view
NeRF	20.32	15.00	12.41	25.32	24.34	21.34
metaNeRF-DTU	22.23	22.07	22.30	20.84	21.12	20.96
pixelNeRF-DTU	22.12	22.09	22.06	19.46	19.87	19.54
DS-NeRF	**10.41**	**8.61**	**8.15**	11.42	10.43	9.43
DS-NeRF w/ RGBD	-	-	-	**5.81**	**5.31**	**4.22**

Table 2.4: **Depth Error.** We compare rendered depth to reference "ground-truth" depth obtained from NeRF Real and Redwood RGB-D. DS-NeRF is able to extract better geometry as indicated by the lower depth errors from test views. We also show DS-NeRF trained with Redwood's dense supervision can significantly improve NeRF's ability to model the underlying geometry.

methods are trained on DTU scenes and therefore can fully leverage dataset priors.

NeRF Real. As seen in Table 2.1, our approach renders images with better scores than than NeRF and LLFF, especially when only two and fives input views are available. We also find that metaNeRF-DTU and pixelNeRF struggle which highlights their apparent weakness. These DTU-pretrained models struggle to perform well outside of DTU. Our full approach is capable of achieving good rendering results because we do not utilize assumptions on the test scene's structure. We also add our depth supervision loss to other methods like pixelNeRF and IBRNet and find their performances improve, showing that many methods can benefit from adding depth supervision. MVSNeRF has an existing geometry prior handled by its PSV-initialization, thus we did not see an improvement from adding depth supervision.

Redwood. Like NeRF Real, we find similar improvements in performance across the Redwood dataset in Table 2.3. Because Redwood includes depth measurements collected with a sensor, we also consider how alternative sources of depth supervision can improve results. We train DS-NeRF, replacing COLMAP supervision with the scaled Redwood depth measurements and find that the denser depth helps even more, achieving a PSNR of 20.3 on 2-views.

2.4.4 Depth error

We evaluate NeRF's rendered depth by comparing them to reference "ground truth" depth measurements. For NeRF Real, we use reference depth of test keypoints recovered from running an all-view dense stereo reconstruction. For Redwood [33], we align their released 3D models with our cameras by running 3dMatch [276] and generate reference depths for each test view. Please refer to our arXiv version for more details regarding depth error evaluation. As shown in Table 2.4, DS-NeRF, trained with supervision obtained only from depth in training views, is able to estimate depth more accurately than all the other NeRF models. While this is not particularly surprising, it does highlight the weakness of training NeRFs only using RGB supervision. For example, in Figure 2.5, NeRF tends to ignore geometry and fails to produce any coherent depth map.

RBG-D inputs. We consider a variant of depth supervision using RGB-D input from Redwood. We derive dense depth map for each training view using 3DMatch [276] with RGB-D input. With dense depth supervision, we can render rays for any pixel in the valid region, and apply our KL depth-supervision loss. As shown in Table 2.3 and Table 2.4, dense depth supervision produces even better-quality images and significantly lower depth errors.

2.4.5 Analysis

Overfitting. Figure 2.2 shows that NeRF can overfit to a small number of input views by learning degenerate 3D geometries. Adding depth supervision can assist NeRF to disambiguate geometry and render better novel views.

Faster Training. To quantify speed improvements in NeRF training, we compare training DS-NeRF and NeRF under identical settings. Like in Section 2.4.3, we evaluate view synthesis quality on test views under various number of input views from NeRF Real using PSNR. We can compare training speed performance by plotting PSNR on test views versus training iterations in Figure 2.7.

DS-NeRF achieves a particular test PSNR threshold using 2-3x less training iterations than NeRF. These benefits are significantly magnified when given fewer views. In the extreme case of only 2-views, NeRF is completely unable to match

DS-NeRF's performance. While these results are given in terms of training iteration, we can translate them into wall time improvements. On a single RTX A5000, a training loop of DS-NeRF takes $\sim$ 362.4 ms/iter while NeRF needs $\sim$ 359.8 ms/iter. Thus in the 5-view case, DS-NeRF achieves NeRF's peak test PSNR around 13 hours faster, a massive improvement considering the negligible cost.

2.5 Discussion.

We introduce Depth-supervised NeRF, a model for learning neural radiance fields that takes advantage of depth supervision. Our model uses "free" supervision provided by sparse 3D point clouds computed during standard SFM pre-processing steps. This additional supervision has a significant impact; DS-NeRF trains 2-3x faster and produces better results from fewer training views (improving PSNR from 13.5 to 20.2). While recent research has sought to improve NeRF by exploiting priors learned from category-specific training data, our approach requires no training and thus generalizes (in principle) to any scenes on which SFM succeeds. This allows us to integrate depth supervision to many NeRF-based methods and observe significant benefits. Finally, we provide cursory experiments that explore alternate forms of depth supervision such as active depth sensors.

Part II

3D Asset Generation

Chapter 3

3D-aware Conditional Image Synthesis

While Chapter 2 demonstrated that depth supervision can dramatically reduce the view requirements for 3D reconstruction, enabling scene capture from as few as two images. However, these approaches are inherently limited to digitizing what already exists in the physical world, offering no path for creating novel 3D content that exists only in imagination.

This limitation motivates a shift from reconstruction to generation: What if we want to create entirely new 3D scenes and objects from scratch? However, as discussed in the challenges section, 3D generation presents its own unique difficulties, particularly around multi-view consistency and the requirement for expensive 3D training data.

The following chapter explores how we can train a 3D-aware generative model leveraging only posed 2D images to enable controllable generation of new 3D objects.

3.1 Introduction

Content creation with generative models has witnessed tremendous progress in recent years, enabling high-quality, user-controllable image and video synthesis [45, 54, 64, 86]. In particular, image-to-image translation methods [75, 174, 295]

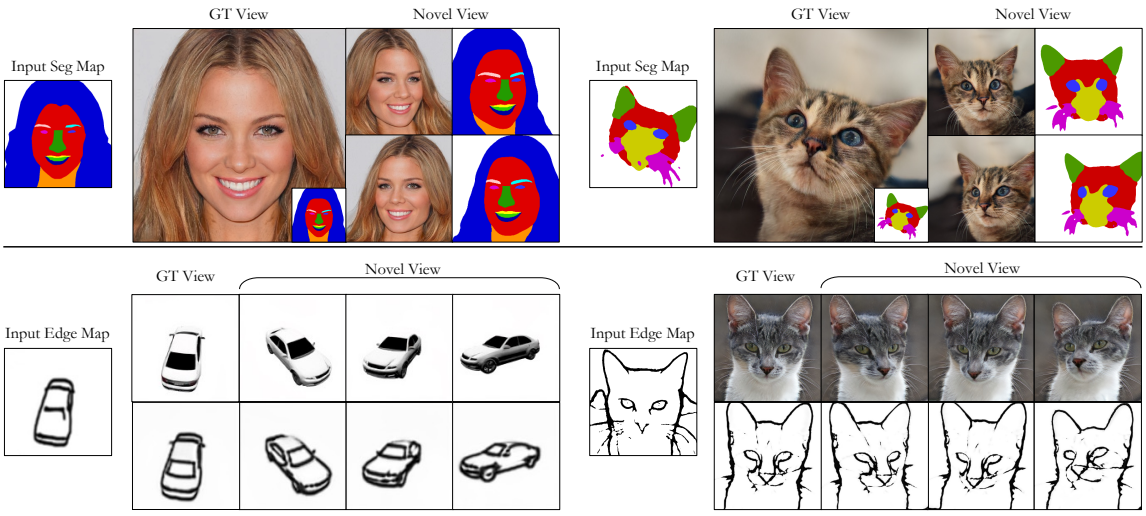


Figure 3.1: Given a 2D label map as input, such as a segmentation or edge map, our model learns to predict high-quality 3D labels, geometry, and appearance, which enables us to render both labels and RGB images from different viewpoints. The inferred 3D labels further allow interactive editing of label maps from any viewpoint, as shown in Figure 3.10.

allow users to interactively create and manipulate a high-resolution image given a 2D input label map. Unfortunately, existing image-to-image translation methods operate purely in 2D, without explicit reasoning of the underlying 3D structure of the content. As shown in Figure 3.1, we aim to make conditional image synthesis 3D-aware, allowing not only 3D content generation but also viewpoint manipulation and attribute editing (e.g., car shape) in 3D.

Synthesizing 3D content conditioned on user input is challenging. For model training, it is costly to obtain large-scale datasets with paired user inputs and their desired 3D outputs. During test time, 3D content creation often requires multi-view user inputs, as a user may want to specify the details of 3D objects using 2D interfaces from different viewpoints. However, these inputs may not be 3D-consistent, providing conflicting signals for 3D content creation.

To address the above challenges, we extend conditional generative models with 3D neural scene representations. To enable *cross-view* editing, we additionally encode semantic information in 3D, which can then be rendered as 2D label maps from different viewpoints. We learn the aforementioned 3D representation using

only 2D supervision in the form of image reconstruction and adversarial losses. While the reconstruction loss ensures the alignment between 2D user inputs and corresponding 3D content, our pixel-aligned conditional discriminator encourages the appearance and labels to look plausible while remaining pixel-aligned when rendered into novel viewpoints. We also propose a cross-view consistency loss to enforce the latent codes to be consistent from different viewpoints.

We focus on 3D-aware semantic image synthesis on the CelebAMask-HQ [99], AFHQ-cat [34], and shapenet-car [19] datasets. Our method works well for various 2D user inputs, including segmentation maps and edge maps. Our method outperforms several 2D and 3D baselines, such as Pix2NeRF variants [13], SofGAN [20], and SEAN [298]. We further ablate the impact of various design choices and demonstrate applications of our method, such as cross-view editing and explicit user control over semantics and style.

3.2 Related Work

Neural Implicit Representation. Neural implicit fields, such as DeepSDF and NeRFs [149, 170], model the appearance of objects and scenes with an implicitly defined, continuous 3D representation parameterized by neural networks. They have produced significant results for 3D reconstruction [214, 299] and novel view synthesis applications [110, 134, 139, 153, 280] thanks to their compactness and expressiveness. NeRF and its descendants aim to optimize a network for an individual scene, given hundreds of images from multiple viewpoints. Recent works further reduce the number of training views through learning network initializations [21, 221, 273], leveraging auxiliary supervision [39, 78], or imposing regularization terms [163]. Recently, explicit or hybrid representations of radiance fields [22, 153, 196] have also shown promising results regarding quality and speed. In our work, we use hybrid representations for modeling both user inputs and outputs in 3D, focusing on synthesizing novel images rather than reconstructing an existing scene. A recent work Pix2NeRF [13] aims to translate a single image to a neural radiance field, which allows single-image novel view synthesis. In contrast, we focus on 3D-aware user-controlled content generation.

Conditional GANs. Generative adversarial networks (GANs) learn the distribution of natural images by forcing the generated and real images to be indistinguishable. They have demonstrated high-quality results on 2D image synthesis and manipulation [1, 7, 12, 54, 86, 87, 88, 177, 203, 232, 293, 294]. Several methods adopt image-conditional GANs [75, 150] for user-guided image synthesis and editing applications [70, 72, 99, 115, 174, 175, 200, 248, 295, 298] . In contrast, we propose a 3D-aware generative model conditioned on 2D user inputs that can render view-consistent images and enable interactive 3D editing. Recently, SoFGAN [20] uses a 3D semantic map generator and a 2D semantic-to-image generator to enable 3D-aware generation, but using 2D generators does not ensure 3D consistency.

3D-aware Image Synthesis. Early data-driven 3D image editing systems can achieve various 3D effects but often require a huge amount of manual effort [27, 91]. Recent works have integrated the 3D structure into learning-based image generation pipelines using various geometric representations, including voxels [61, 297], voxelized 3D features [159], and 3D morphable models [227, 269]. However, many rely on external 3D data [227, 269, 297]. Recently, neural scene representations have been integrated into GANs to enable 3D-aware image synthesis [17, 18, 55, 162, 166, 168, 202, 266]. Intriguingly, these 3D-aware GANs can learn 3D structures without any 3D supervision. For example, StyleNeRF [55] and EG3D [18] learn to generate 3D representations by modulating either NeRFs or explicit representations with latent style vectors. This allows them to render high-resolution view-consistent images. Unlike the above methods, we focus on conditional synthesis and interactive editing rather than random sampling. Several works [38, 74, 131, 260] have explored sketch-based shape generation but they do not allow realistic image synthesis.

Closely related to our work, Huang et al. [67] propose synthesizing novel views conditional on a semantic map. Our work differs in three ways. First, we can predict full 3D labels, geometry, and appearance, rather than only 2D views, which enables cross-view editing. Second, our method can synthesize images with a much wider baseline than Huang et al. [67]. Finally, our learning algorithm does not require ground truth multi-view images of the same scene. Two recent works, FENeRF [216] and 3DSGAN [279], also leverage semantic labels for training 3D-aware GANs, but they do not support conditional inputs and require

additional efforts (e.g., GAN-inversion) to allow user editing. Three concurrent works, IDE-3D [215], NeRFFaceEditing [80], and sem2nerf [31], also explore the task of 3D-aware generation based on segmentation masks. However, IDE-3D and sem2nerf only allow editing on a fixed view, and NeRFFaceEditing focuses on real image editing rather than generation. All of them do not include results for other input modalities. In contrast, we present a general-purpose method that works well for diverse datasets and input controls.

3.3 Method

Given a 2D label map $\mathbf{I_s}$, such as a segmentation or edge map, `pix2pix3D` generates a 3D-volumetric representation of geometry, appearance, and labels that can be rendered from different viewpoints. Figure 3.2 provides an overview. We first introduce the formulation of our 3D conditional generative model for 3D-aware image synthesis in Section 3.3.1. Then, in Section 3.3.2, we discuss how to learn the model from color and label map pairs $\{\mathbf{I_c}, \mathbf{I_s}\}$ associated with poses $\mathbf{P}$.

3.3.1 Conditional 3D Generative Models

Similar to EG3D [18], we adopt a hybrid representation for the density and appearance of a scene and use style vectors to modulate the 3D generations. To condition the 3D representations on 2D label map inputs, we introduce a conditional encoder that maps a 2D label map into a latent style vector. Additionally, `pix2pix3D` produces 3D labels that can be rendered from different viewpoints, allowing for cross-view user editing.

Conditional Encoder. Given a 2D label map input $\mathbf{I_s}$ and a random latent code sampled from the spherical Gaussian space $\mathbf{z} \sim \mathcal{N}(0, I)$, our conditional encoder E outputs a list of style vectors $\mathbf{w}^+ \in \mathbb{R}^{l \times 256}$,

$$\mathbf{w}^+ = E(\mathbf{I_s}, \mathbf{z}),$$

where $l = 13$ is the number of layers to be modulated. Specifically, we encode $\mathbf{I_s}$ into the first 7 style vectors that represent the global geometric information of the

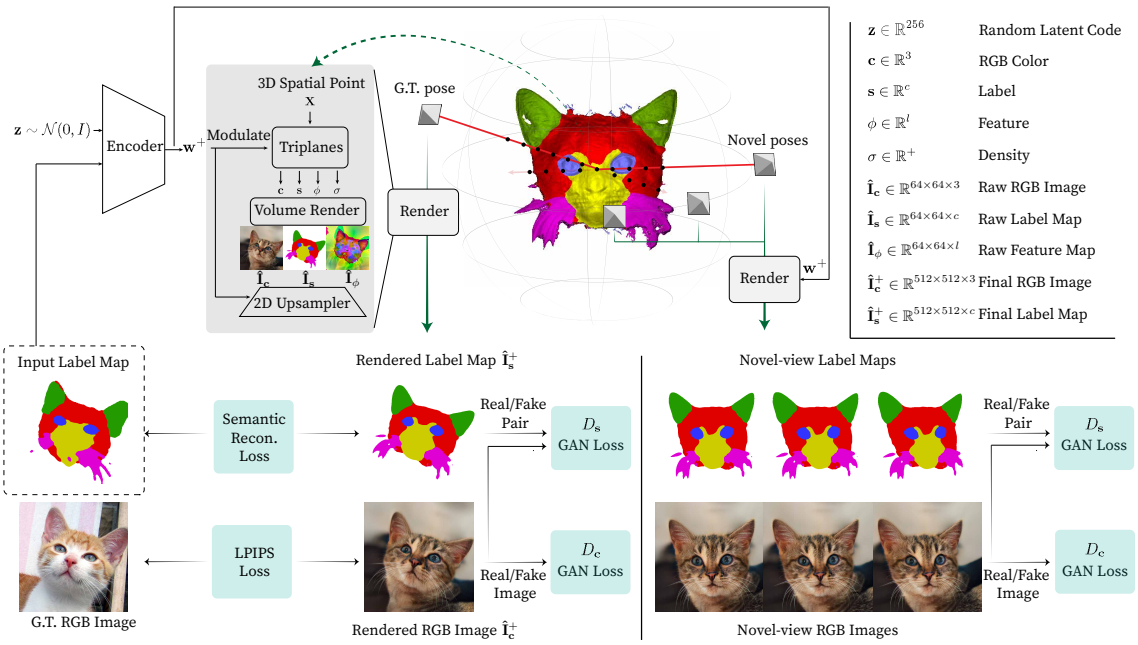


Figure 3.2: **Overall pipeline.** Given a 2D label map (e.g., segmentation map), a random latent code z, and a camera pose $\hat{P}$ as inputs, our generator renders the label map and image from viewpoint $\hat{P}$. Intuitively, the input label map specifies the geometric structure, while the latent code captures the appearance, such as hair color. We begin with an encoder that encodes both the input label map and the latent code into style vectors $\mathbf{w}^+$. We then use $\mathbf{w}^+$ to modulate our 3D representation, which takes a spatial point $\mathbf{x}$ and outputs (1) color $\mathbf{c} \in \mathbb{R}^3$, (2) density σ, (3) feature $\phi \in \mathbb{R}^l$, and (4) label $\mathbf{s} \in \mathbb{R}^c$. We then perform volumetric rendering and 2D upsampling to get the high-res label map $\hat{\mathbf{I}}_\mathbf{s}^+$ and RGB Image $\hat{\mathbf{I}}_\mathbf{c}^+$. For those rendered from ground-truth poses, we compare them to ground-truth labels and images with an LPIPS loss and label reconstruction loss. We apply a GAN loss on labels and images rendered from both novel and original viewpoints.

scene. We then feed the random latent code $\mathbf{z}$ through a Multi-Layer Perceptron (MLP) mapping network to obtain the rest of the style vectors that control the appearance.

Conditional 3D Representation. Our 3D representation is parameterized by tri-planes followed by an 2-layer MLP f [18], which takes in a spatial point $\mathbf{x} \in \mathbb{R}^3$ and returns 4 types of outputs: (1) color $\mathbf{c} \in \mathbb{R}^3$, (2) density $\sigma \in \mathbb{R}^+$, (3) feature $\phi \in \mathbb{R}^{64}$ for the purpose of 2D upsampling, and most notably, (4) label $\mathbf{s} \in \mathbb{R}^c$, where c is the number of classes if $\mathbf{I_s}$ is a segmentation map, otherwise 1 for edge labels. We make the field conditional by modulating the generation of tri-planes F^{tri} with the style vectors $\mathbf{w}^+$. We also remove the view dependence of the color following [18, 55]. Formally,

$$(\mathbf{c}, \mathbf{s}, \sigma, \phi) = f(F^{\text{tri}}_{\mathbf{w}^+}(\mathbf{x})).$$

Volume Rendering and Upsampling. We apply volumetric rendering to synthesize color images [84, 149]. In addition, we render label maps, which are crucial for enabling cross-view editing (Section 3.4.3) and improving rendering quality (Table 3.1). Given a viewpoint $\hat{P}$ looking at the scene origin, we sample N points along the ray that emanates from a pixel location and query density, color, labels, and feature information from our 3D representation. Let $\mathbf{x_i}$ be the i-th sampled point along the ray r. Let $\mathbf{c}_i, \mathbf{s}_i$ and $\mathbf{Œ}_i$ be the color, labels, and the features of $\mathbf{x_i}$. Similar to [216], The color, label map, and feature images are computed as the weighted combination of queried values,

$$\hat{\mathbf{I}}_{\mathbf{c}}(r) = \sum_{i=1}^{N} \tau_i \mathbf{c}_i, \quad \hat{\mathbf{I}}_{\mathbf{s}}(r) = \sum_{i=1}^{N} \tau_i \mathbf{s}_i, \quad \hat{\mathbf{I}}_{\mathbf{Œ}}(r) = \sum_{i=1}^{N} \tau_i \mathbf{Œ}_i, \tag{3.1}$$

where the transmittance τ_i is computed as the probability of a photon traversing between the camera center and the i-th point given the length of the i-th interval δ_i,

$$\tau_i = \prod_{j=1}^{i} \exp\left(-\sigma_j \delta_j\right)\left(1 - \exp\left(-\sigma_i \delta_i\right)\right).$$

Similar to prior works [18, 55, 166], we approximate Equation 3.1 by 2D Upsampler

U to reduce the computational cost. We render high-res 512×512 images in two passes. In the first pass, we render low-res 64×64 images $\hat{\mathbf{I}}_{\mathbf{c}}, \hat{\mathbf{I}}_{\mathbf{s}}, \hat{\mathbf{I}}_{\mathbf{\Phi}}$. Then a CNN up-sampler U is applied to obtain high-res images,

$$\hat{\mathbf{I}}_{\mathbf{c}}^{+} = U(\hat{\mathbf{I}}_{\mathbf{c}}, \hat{\mathbf{I}}_{\mathbf{\Phi}}), \qquad \hat{\mathbf{I}}_{\mathbf{s}}^{+} = U(\hat{\mathbf{I}}_{\mathbf{s}}, \hat{\mathbf{I}}_{\mathbf{\Phi}}).$$

3.3.2 Learning Objective

Learning conditional 3D representations from monocular images is challenging due to its under-constrained nature. Given training data of associated images, label maps, and camera poses predicted by an off-the-shelf model, we carefully construct learning objectives, including reconstruction, adversarial, and cross-view consistency losses. These objectives will be described below.

Reconstruction Loss. Given a ground-truth viewpoint $\mathbf{P}$ associated with the color and label maps $\{\mathbf{I}_{\mathbf{c}}, \mathbf{I}_{\mathbf{s}}\}$, we render color and label maps from $\mathbf{P}$ and compute reconstruction losses for both high-res and low-res output. We use LPIPS [284] to compute the image reconstruction loss $\mathcal{L}_c$ for color images. For label reconstruction loss $\mathcal{L}_s$, we use the balanced cross-entropy loss for segmentation maps or L2 Loss for edge maps,

$$\mathcal{L}_{\text{recon}} = \lambda_c \mathcal{L}_c(\mathbf{I}_{\mathbf{c}}, \{\hat{\mathbf{I}}_{\mathbf{c}}, \hat{\mathbf{I}}_{\mathbf{c}}^{+}\}) + \lambda_s \mathcal{L}_s(\mathbf{I}_{\mathbf{s}}, \{\hat{\mathbf{I}}_{\mathbf{s}}, \hat{\mathbf{I}}_{\mathbf{s}}^{+}\}),$$

where λ_c and λ_s balance two terms.

Pixel-aligned Conditional Discriminator. The reconstruction loss alone fails to synthesize detailed results from novel viewpoints. Therefore, we use an adversarial loss [54] to enforce renderings to look realistic from random viewpoints. Specifically, we have two discriminators $D_{\mathbf{c}}$ and $D_{\mathbf{s}}$ for RGB images and label maps, respectively. $D_{\mathbf{c}}$ is a widely-used GAN loss that takes real and fake images as input, while the pixel-aligned conditional discriminator $D_{\mathbf{s}}$ concatenates color images and label maps as input, which encourages pixel alignment between color images and label maps. Notably, in $D_{\mathbf{s}}$, we stop the gradients for the color images to prevent a potential quality downgrade. We also feed the rendered low-res images to prevent the upsampler from hallucinating details, inconsistent with the low-res output.

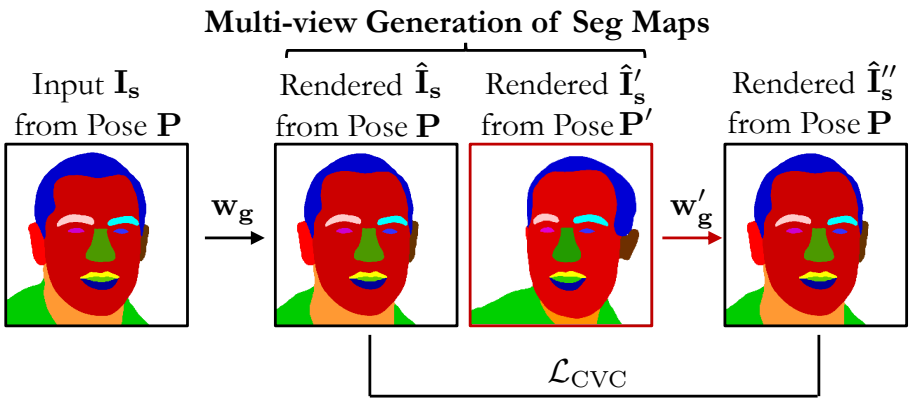


Figure 3.3: **Cross-View Consistency Loss.** Given an input label map $\mathbf{I_s}$ and its associated pose $\mathbf{P}$, we first infer the geometry latent code $\mathbf{w_g}$. From $\mathbf{w_g}$, we can generate a label map $\hat{\mathbf{I}}_\mathbf{s}$ from the same pose $\mathbf{P}$, and $\hat{\mathbf{I}}'_\mathbf{s}$ from a random pose $\mathbf{P}'$. Next, we infer $\mathbf{w}'_\mathbf{g}$ from the novel view $\hat{\mathbf{I}}'_\mathbf{s}$, and render it back to the original pose $\mathbf{P}$ to obtain $\hat{\mathbf{I}}''_\mathbf{s}$. Finally, we add a reconstruction loss: $\mathcal{L}_{\text{CVC}} = \lambda_{\text{CVC}}\mathcal{L}_s(\hat{\mathbf{I}}''_\mathbf{s}, \hat{\mathbf{I}}_\mathbf{s})$.

The adversarial loss can be written as follows.

$$\mathcal{L}_{\text{GAN}} = \lambda_{D_\mathbf{c}}\mathcal{L}_{D_\mathbf{c}}(\hat{\mathbf{I}}^+_\mathbf{c}, \hat{\mathbf{I}}_\mathbf{c}) + \lambda_{D_\mathbf{s}}\mathcal{L}_{D_\mathbf{s}}(\hat{\mathbf{I}}^+_\mathbf{c}, \hat{\mathbf{I}}_\mathbf{c}, \hat{\mathbf{I}}^+_\mathbf{s}, \hat{\mathbf{I}}_\mathbf{s}).$$

where $\lambda_{D_\mathbf{c}}$ and $\lambda_{D_\mathbf{s}}$ balance two terms. To stabilize the GAN training, we adopt the R1 regularization loss [141].

Cross-view Consistency Loss. We observe that inputting label maps of the same object from different viewpoints will sometimes result in different 3D shapes. Therefore we add a cross-view consistency loss to regularize the training, as illustrated in Figure 3.3. Given an input label map $\mathbf{I_s}$ and its associated pose $\mathbf{P}$, we generate the label map $\hat{\mathbf{I}}'_\mathbf{s}$ from a different viewpoint $\mathbf{P}'$, and render the label map $\hat{\mathbf{I}}''_\mathbf{s}$ back to the pose $\mathbf{P}$ using $\hat{\mathbf{I}}'_\mathbf{s}$ as input. We add a reconstruction loss between $\hat{\mathbf{I}}''_\mathbf{s}$ and $\hat{\mathbf{I}}_\mathbf{s}$:

$$\mathcal{L}_{\text{CVC}} = \lambda_{\text{CVC}}\mathcal{L}_s(\hat{\mathbf{I}}''_\mathbf{s}, \hat{\mathbf{I}}_\mathbf{s}),$$

where $\mathcal{L}_s$ denotes the reconstruction loss in the label space, and λ_{CVC} weights the loss term. This loss is crucial for reducing error accumulation during cross-view editing.

Figure 3.4: **Qualitative Comparison with Pix2NeRF [13], SoFGAN [20], and SEAN [298]** on CelebAMask dataset for seg2face task. SEAN fails in multi-view synthesis, while SoFGAN suffers from multi-view inconsistency (e.g., face identity changes across viewpoints). Our method renders high-quality images while maintaining multi-view consistency.

Optimization. Our final learning objective is written as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{GAN}} + \mathcal{L}_{\text{CVC}}.$$

At every iteration, we determine whether to use a ground-truth pose or sample a random one with a probability of p. We use the reconstruction loss and GAN loss for ground-truth poses, while for random poses, we only use the GAN loss.

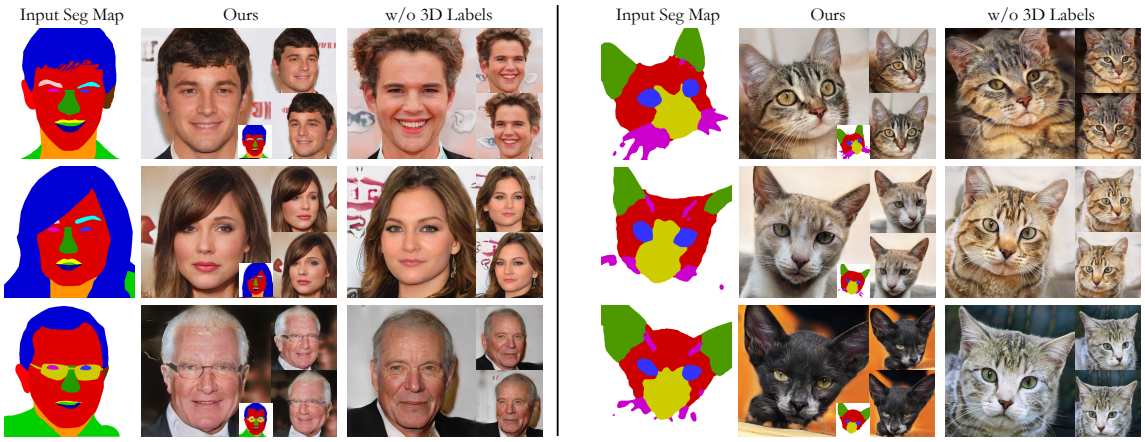


Figure 3.5: **Qualitative ablation on seg2face and seg2cat.** We ablate our method by removing the branch that renders label maps (*w/o 3D Labels*). Our results better align with input labels (e.g., hairlines and the cat's ear).

Seg2Face	Quality			Alignment		
			SG			FVV
CelebAMask	FID ↓	KID ↓	Diversity ↑	mIoU ↑	acc ↑	Identity ↓
SEAN	32.74	0.018	0.29	0.52	0.85	N/A
SoFGAN	23.34	0.012	**0.33**	**0.53**	0.89	0.58
Pix2NeRF	54.23	0.042	0.16	0.36	0.65	0.44
`pix2pix3D` **(Ours)**						
w/o 3D Labels	12.96	0.005	0.30	N/A (0.43)	N/A (0.81)	0.38
w/o CVC	11.62	0.004	0.30	0.50 (0.50)	0.87 (0.85)	0.42
full model	11.54	**0.003**	0.28	0.51 (0.52)	**0.90** (0.88)	**0.36**
full model†	**11.13**	**0.003**	0.29	0.51 (0.50)	**0.90** (0.87)	**0.36**

Table 3.1: **Seg2face Evaluation.** Our metrics include image quality (FID, KID, SG Diversity), alignment (mIoU and acc against GT label maps), and multi-view consistency (FVV Identity). Single-generation diversity (SG Diversity) is obtained by computing the LPIPS metric between randomly generated pairs given a single conditional input. To evaluate alignment, we compare the generated label maps against the ground truth in terms of mIoU and pixel accuracy (acc). Alternatively, given a generated image, one could estimate label maps via a face parser, and compare those against the ground truth (numbers in parentheses). We include SEAN [298] and SoFGAN [20] as baselines, and modify Pix2NeRF [13] to take conditional input. Our method achieves the best quality, alignment ACC, and FVV Identity while being competitive on SG Diversity. SoFGAN tends to have better alignment but worse 3D consistency. We also ablate our method w.r.t the 3D labels and the cross-view consistency (CVC) loss. Our 3D labels are crucial for alignment, while the CVC loss improves multi-view consistency. Using pretrained models from EG3D (†) also improves the performance.

3.4 Experiment

We first introduce the datasets and evaluation metrics. Then we compare our method with the baselines. Finally, we demonstrate cross-view editing and multi-modal synthesis applications enabled by our method.

Datasets. We consider four tasks: *seg2face*, *seg2cat*, *edge2cat*, and *edge2car* in our ex-

Edge2Car	Quality			Alignment
	FID ↓	KID ↓	SG Diversity ↑	AP ↑
Pix2NeRF	23.42	0.014	0.06	0.28
`pix2pix3D` **(Ours)**				
w/o 3D Labels	10.73	0.005	0.12	0.45 (0.42)
w/o CVC	9.42	**0.004**	**0.13**	0.61 (0.59)
full model	**8.31**	**0.004**	**0.13**	**0.63** (0.59)

Table 3.2: **Edge2car Evaluation.** We compare our method with Pix2NeRF [13] on edge2car using the shapenet-car [19] dataset. Similar to Table 3.1, we evaluate FID, KID, and SG Diversity for image quality. We also evaluate the alignment with the input edge map using AP. Similarly, we can either run informative drawing [16] on generated images to obtain edge maps (numbers in parentheses) or directly use generated edge maps to calculate the metrics. We achieve better image quality and alignment than Pix2NeRF. We also find that using 3D labels and cross-view consistency loss is helpful regarding FID and AP metrics.

periments. For seg2face, we use CelebAMask-HQ [99] for evaluation. CelebAMask-HQ contains 30,000 high-resolution face images from CelebA [126], and each image has a facial part segmentation mask and a predicted pose. The segmentation masks contain 19 classes, including skin, eyebrows, ears, mouth, lip, etc. The pose associated with each image segmentation is predicted by HopeNet [193]. We split the CelebAMask-HQ dataset into a training set of 24,183, a validation set of 2,993, and a test set of 2,824, following the original work [99]. For seg2cat and edge2cat, we use AFHQ-cat [34], which contains 5,065 images at 512× resolution. We estimate the viewpoints using unsup3d [256]. We extract the edges using pidinet [213] and obtain segmentation by clustering DINO features [3] into 6 classes. For edge2car, we use 3D models from shapenet-car [19] and render 500,000 images at 128× resolution for training, and 30,000 for evaluation. We extract the edges using informative drawing [16]. We train our model at 512× resolution except for 128× in the edge2car task.

Running Time. For training the model at 512× resolution, it takes about three days on eight RTX 3090 GPUs. But we can significantly reduce the training time to

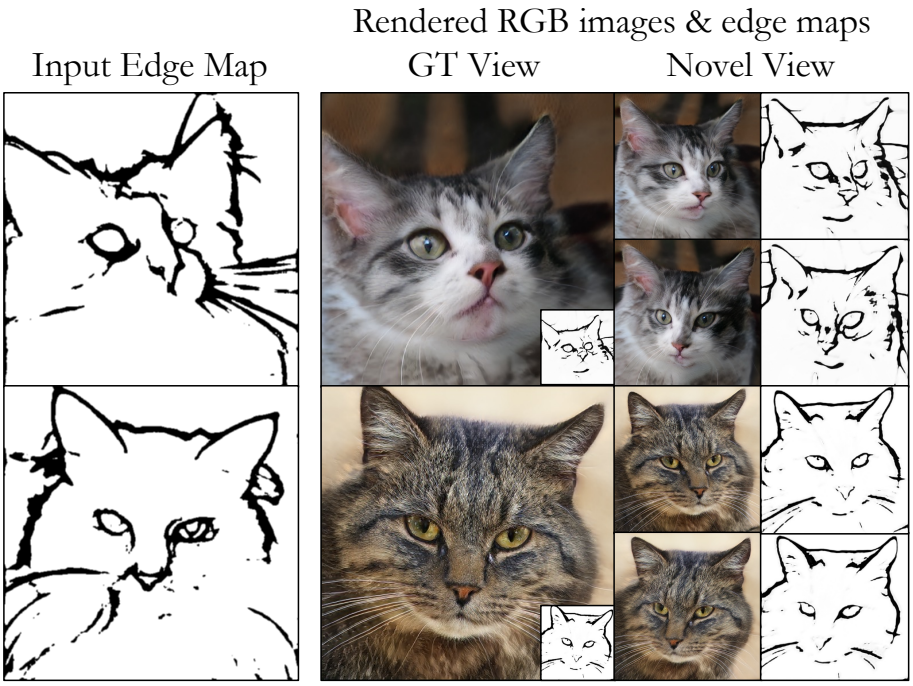


Figure 3.6: **Results on edge2cat.** Our model is trained on AFHQ-cat [34] with edges extracted by pidinet [213].

4 hours if we initialize parts of our model with pretrained weights from EG3D[18]. During inference, our model takes 10 ms to obtain the style vector, and another 30 ms to render the final image and the label map on a single RTX A5000. The low latency (25 FPS) allows for interactive user editing.

3.4.1 Evaluation metrics

We evaluate the models from two aspects: 1) the image quality regarding fidelity and diversity, and 2) the alignment between input label maps and generated outputs.

Quality Metrics. Following prior works [55, 173], we use the `clean-fid` library [176] to compute Fréchet Inception Distance (FID) [63] and Kernel Inception Distance (KID) [10] to measure the distribution distance between synthesized results and real images. We also evaluate the single-generation diversity (SG Diversity) by calculating the LPIPS metric between randomly generated pairs given a single input following prior works [20, 296]. For FID and KID, we generate 10 images per

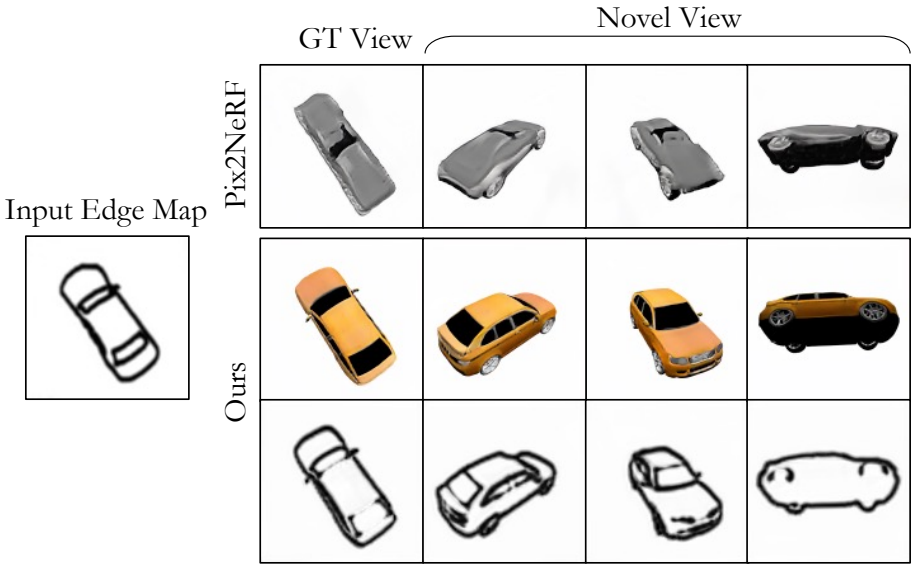


Figure 3.7: **Qualitative comparisons on edge2car.** `pix2pix3D` (Ours) and Pix2NeRF [13] are trained on shapenet-car [19], and `pix2pix3D` achieves better quality and alignment than Pix2NeRF.

label map in the test set using randomly sampled z. We compare our generated images with the whole dataset, including training and test images.

Alignment Metrics. We evaluate models on the test set using mean Intersection-over-Union (mIoU) and pixel accuracy (acc) for segmentation maps following existing works [173, 200], and average precision (AP) for edge maps. For those models that render label maps as output, we directly compare them with ground-truth labels. Otherwise, we first predict the label maps from the output RGB images using off-the-shelf networks [99, 213], and then compare the prediction with the ground truth. The metrics regarding such predicted semantic maps are reported within brackets in Table 3.1 and Table 3.2.

For seg2face, we evaluate the preservation of facial identity from different viewpoints (FVV Identity) by calculating their distances with the dlib face recognition algorithm[1].

[1] https://github.com/ageitgey/face_recognition

Seg2Cat	Quality			Alignment	
AFHQ-cat	FID ↓	KID ↓	SG Diversity ↑	mIoU ↑	acc ↑
Pix2NeRF	43.92	0.081	0.15	0.27	0.58
Ours					
w/o 3D Labels	10.41	0.004	0.26	N/A (0.49)	N/A (0.69)
w/o CVC	9.64	0.004	0.26	**0.66** (0.63)	0.76 (0.73)
Full Model	**8.62**	**0.003**	**0.27**	**0.66** (0.62)	**0.78** (0.73)

Table 3.3: **Seg2cat Evaluation.** We compare our method with Pix2NeRF [13] on Seg2Cat using AFHQ-cat dataset [34], with segmentation obtained by clustering DINO features [3]. Similar to Table 3.1, we evaluate the image quality and alignment. Ours performs better in all metrics.

3.4.2 Baseline comparison

Baselines. Since there are no prior works on conditional 3D-aware image synthesis, we make minimum modifications to Pix2NeRF [13] to be conditional on label maps instead of images. For a thorough comparison, we introduce several baselines: SEAN [298] and SoFGAN [20]. 2D baselines like SEAN [298] cannot generate multi-view images by design (N/A for FVV Identity), while SoFGAN [20] uses an unconditional 3D semantic map generator before the 2D generator so we can evaluate FVV Identity for that.

Results. Figure 3.4 shows the qualitative comparison for seg2face and Table 3.1 reports the evaluation results. SoFGAN [20] tends to produce results with slightly better alignment but worse 3D consistency for its 2D RGB generator. Our method achieves the best quality, alignment acc, and FVV Identity while being competitive with 2D baselines on SG diversity. Figure 3.5 shows the qualitative ablation on seg2face and seg2cat. Table 3.4 reports the metrics for seg2cat. Figure 3.6 shows the example results for edge2cat. Figure 3.7 shows the qualitative comparison for edge2car and Table 3.2 reports the metrics. Our method achieves the best image quality and alignment. Figure 3.8 shows semantic meshes of human and cat faces, extracted by marching cubes and colored by our learned 3D labels.

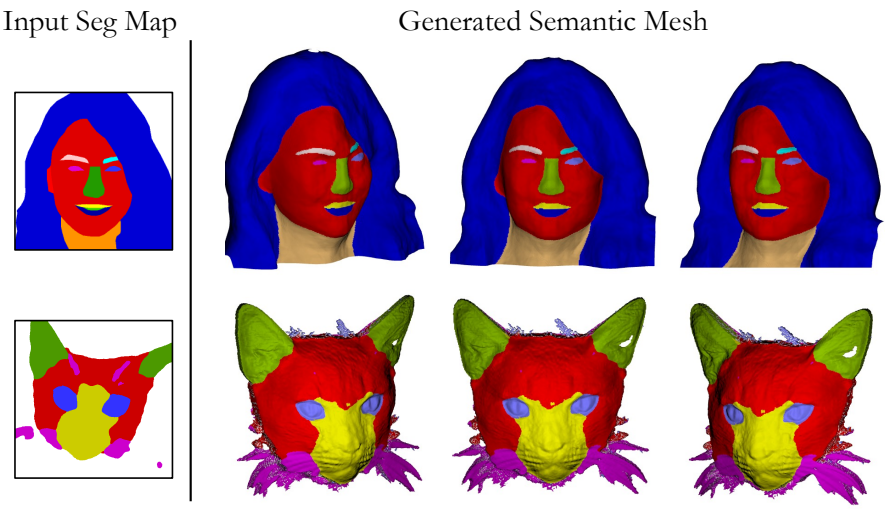


Figure 3.8: **Semantic Mesh.** We show semantic meshes of human and cat faces from marching cubes colored by 3D labels.

Ablation Study. We compare our full method to several variants. Specifically, (1) W/O 3D LABELS, we remove the branch of rendering label maps from our method, and (2) W/O CVC, we remove the cross-view consistency loss. From Table 3.1, Table 3.2, and Figure 3.5, rendering label maps is crucial for the alignment with the input. We posit that the joint learning of appearance, geometry, and label information poses strong constraints on correspondence between the input label maps and the 3D representation. Thus our method can synthesize images pixel-aligned with the inputs. Our CVC loss helps preserve the facial identity from different viewpoints.

Seg2Car	Quality			Alignment	
Shapnet-car	FID ↓	KID ↓	SG Diversity ↑	mIoU ↑	acc ↑
Pix2NeRF	25.86	0.018	0.08	0.24	0.59
Ours	**9.35**	**0.004**	**0.14**	**0.58**	**0.88**

Table 3.4: **Seg2car Evaluation.** We compare our method with Pix2NeRF [13]. Ours performs better in all metrics.

Analysis on random sampling of poses. We study the effect of the different

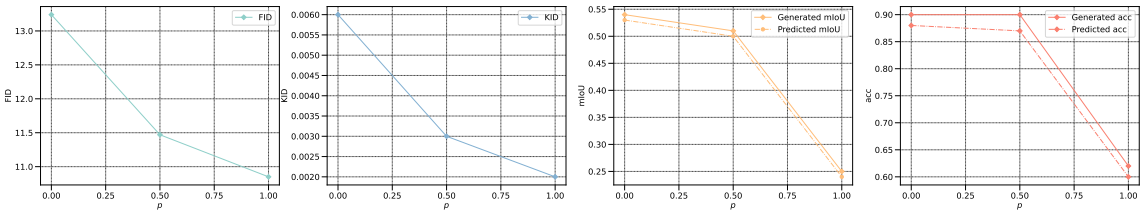


Figure 3.9: We study the effect of random pose sampling probability p during training. Without random poses ($p = 0$), the model achieves the best alignment with input semantic maps, with reduced image quality. In contrast, *only* using random poses ($p = 1$) achieves the best image quality, while results fail to align with input maps. We find $p = 0.5$ balances the image quality and input alignment.

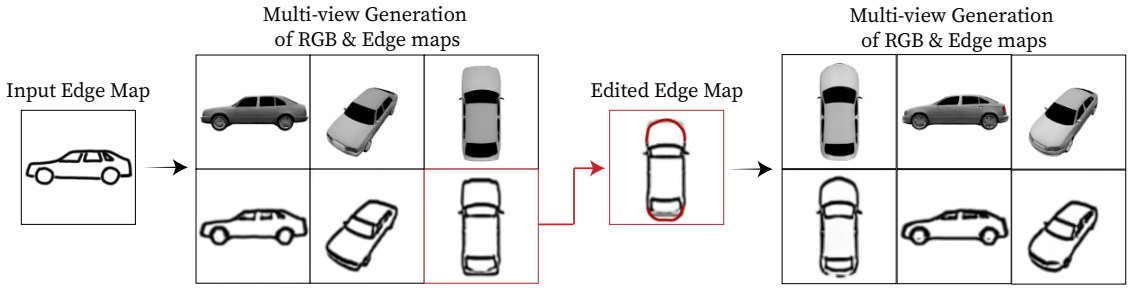


Figure 3.10: **Cross-view Editing of Edge2Car.** Our 3D editing system allows users to edit label maps from any viewpoint instead of only the input view. Importantly, our feed-forward encoder allows fast inference of the latent code without GAN-inversion. Typically, a single forward pass of rendering takes only 40 ms on a single RTX A5000, which enables interactive editing. Please check our demo video on our website.

probabilities of sampling random poses during training, as shown in Figure 3.9. When sampling no random poses ($p = 0$), the model best aligns with input label maps with suboptimal image quality. Conversely, *only* sampling random poses ($p = 1$) gives the best image quality but suffers huge misalignment with input label maps. We find $p = 0.5$ achieves the balance between the image quality and the alignment with the input.

3.4.3 Applications

Cross-view Editing. As shown in Figure 3.10, our 3D editing system allows users to generate and edit label maps from any viewpoint instead of only the input view.

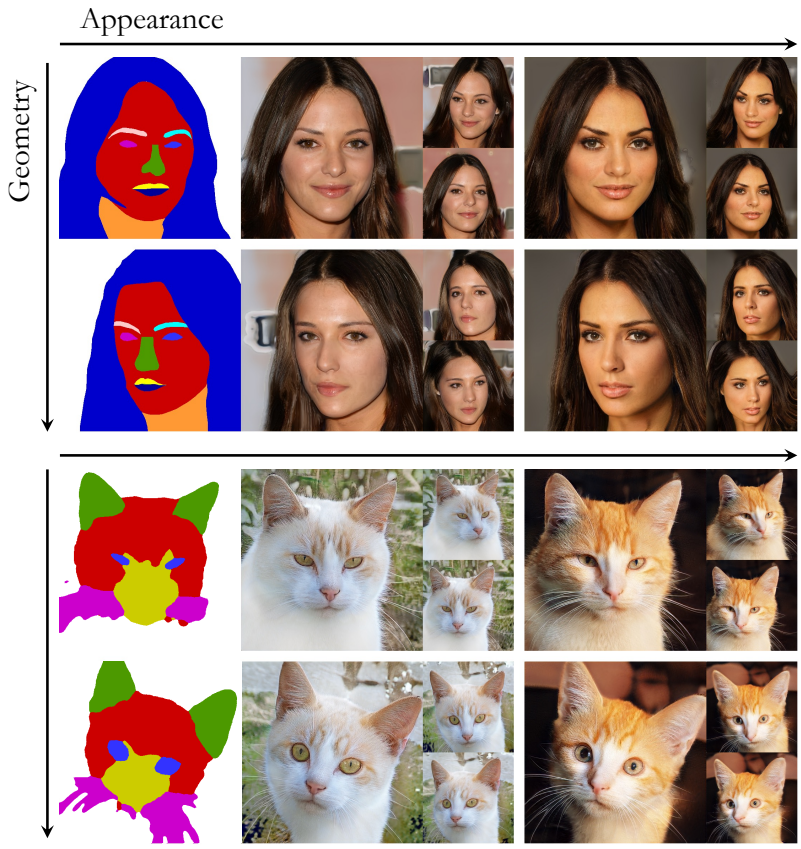


Figure 3.11: **Multi-modal Synthesis.** The leftmost column is the input segmentation map. We use the same segmentation map for each row. We generate multi-modal results by randomly sampling an appearance style for each column.

The edited label map is further fed into the conditional encoder to update the 3D representation. Unlike GAN inversion [294], our feed-forward conditional encoder allows fast inference of the latent code. Thus, a single forward pass of our full model takes only 40 ms on a single RTX A5000.

Multi-modal synthesis and interpolation. Like other style-based generative models [18, 55, 86, 87], our method can disentangle the geometry and appearance information. Specifically, the input label map captures the geometry information while the randomly sampled latent code controls the appearance. We show style manipulation results in Figure 3.11. We can also interpolate both the geometry styles and the appearance styles (Figure 3.12). These results show the clear disentanglement of our 3D representation.

3.5 Discussion

We have introduced `pix2pix3D`, a 3D-aware conditional generative model for controllable image synthesis. Given a 2D label map, our model allows users to render images given any viewpoint. Our model augments the neural field with 3D labels, assigning label, color, and density to every 3D point, allowing for the simultaneous rendering of the image and a pixel-aligned label map. The learned 3D labels further enable interactive 3D cross-view editing. We discuss the broader impact and limitations in the appendix.

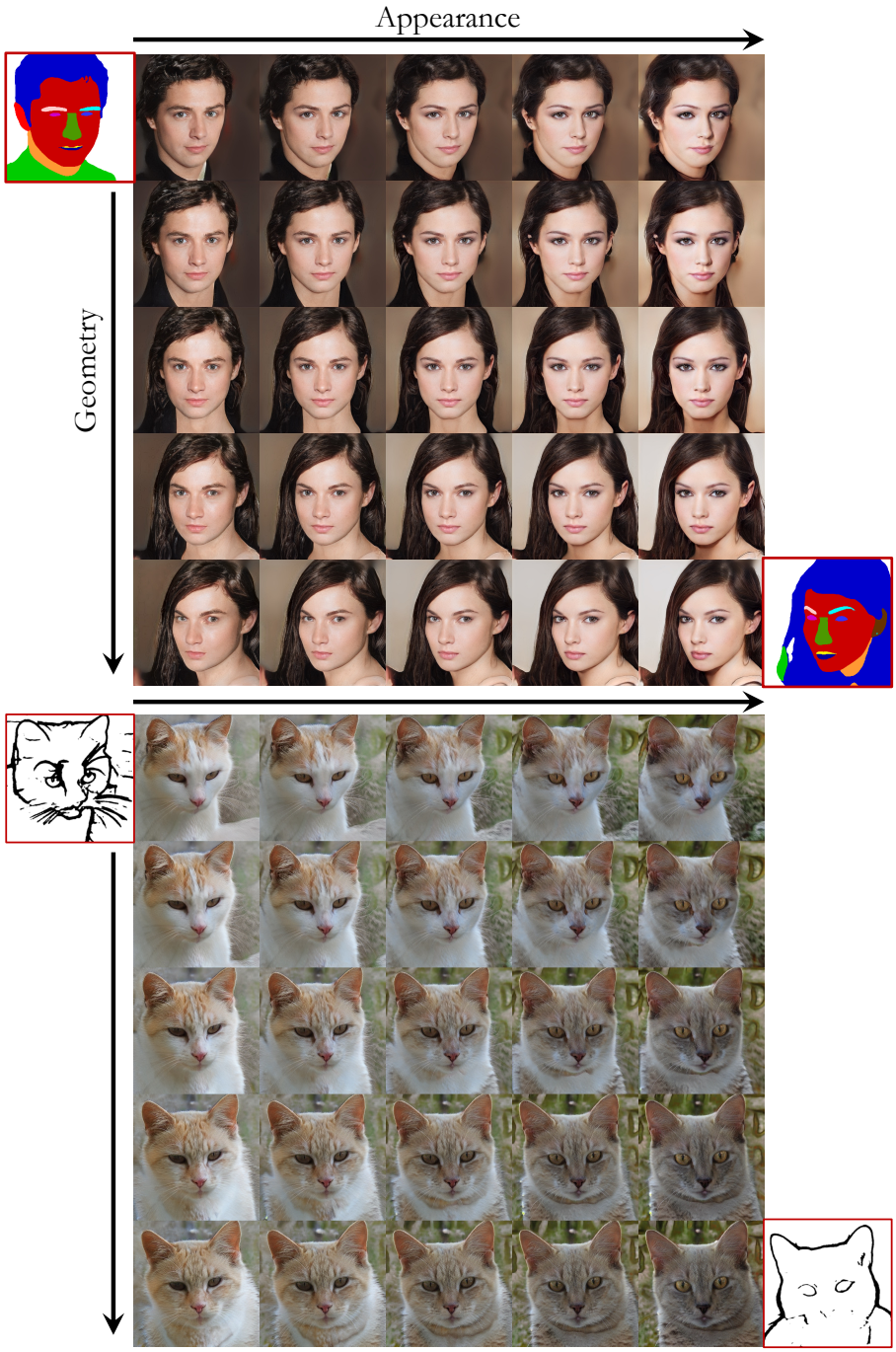


Figure 3.12: **Interpolation.** In each 5×5 grid, the images at the top left and bottom right are generated from the input maps next to them. Each row interpolates two images in label space, while each column interpolates the appearance. For camera poses, we interpolate the pitch along the row and the yaw along the column.

Chapter 4

Efficient Autoregressive Shape Generation via Octree-Based Adaptive Tokenization

Chapter 3 demonstrated that by incorporating 3D awareness into the generation process, we can create editable 3D objects from simple 2D semantic inputs like sketches or segmentation maps. However, this approach is designed for category-specific generation. It works well for cars, faces, or other well-defined object categories with sufficient training data, but cannot easily generalize to the vast diversity of objects and scenes users might want to create. Furthermore, the reliance on 2D semantic inputs, while expressive and intuitive, is not as accessible as simple text descriptions.

The following chapter explores how to scale our training to generic text-to-3D generation that automatically creates arbitrary high-fidelity 3D shapes based on text descriptions.

4.1 Introduction

Recent advances in generative models have revolutionized the field of 3D content creation, enabling diverse applications, including shape generation [106, 142, 235], text-to-3D generation [111, 179, 249], text-driven mesh texturing [23, 41], single-

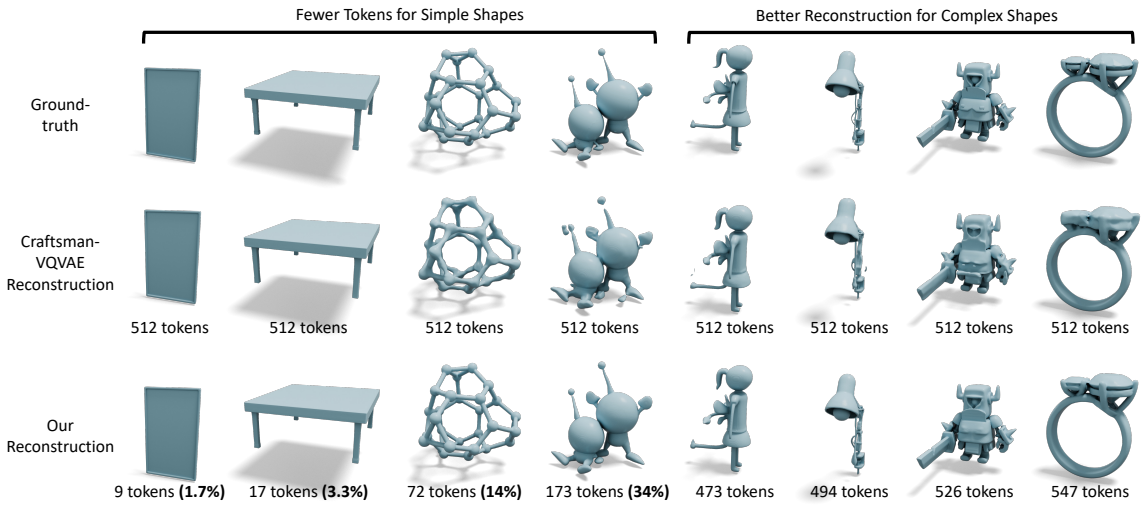


Figure 4.1: We propose an Octree-based Adaptive shape Tokenization (OAT) that dynamically allocates tokens based on shape complexity. Our approach achieves *better* reconstruction quality with *fewer* tokens on average (439 compared to 512 on the full test set) by intelligently distributing more tokens to complex shapes while saving on simpler ones.

image 3D generation [123, 128], and 3D scene editing [59, 124]. One popular paradigm among state-of-the-art methods employs 3D-native diffusion or autoregressive models [106, 188, 261, 282, 288] on top of 3D latents learned from large-scale datasets. As a result, the effectiveness of these models heavily depends on how well 3D shapes are represented and encoded as latent representations.

Effective latent representations for 3D shapes must address several fundamental challenges. First, 3D data is inherently sparse, with meaningful information concentrated primarily on surfaces rather than distributed throughout the volume. Second, real-world objects vary in geometric complexity, ranging from simple primitives to intricate structures with fine details, requiring representation structures that can adapt accordingly. Third, the encoding process must take into account capturing fine local details while preserving the global geometric structure.

Most existing shape VAEs [106, 278, 282, 288] encode shapes into fixed-size latent representations and fail to adapt to the inherent variations in geometric complexity within such shapes. As shown in Figure 4.1 (bottom), objects are encoded with identical latent capacity regardless of their scale, sparsity, or complexity,

resulting in inefficient compression and degraded performance in downstream generative models. While some approaches [188, 261] leverage sparse voxel representations like octrees to account for sparsity, they still subdivide any cell containing surface geometry to the finest level, thus failing to adapt to shape complexity. As illustrated in Figure 4.2, a simple cube with only eight vertices requires similar representation capacity as a highly detailed sculpture in traditional octree structures. Ideally, hierarchical shape representations should adapt to the complexity of different regions within a shape. For instance, in the bottom right of Figure 4.2, complex structures like a tree canopy should require finer subdivision than simpler regions like the trunk.

To address these challenges, we propose an Octree-based Adaptive Tokenization. Our approach dynamically adjusts the latent representation based on local geometric complexity measured by quadric error, efficiently representing both simple and intricate regions with appropriate detail levels. As shown in Figure 4.1, our approach achieves better reconstruction quality with comparable or fewer shape tokens. By developing an Octree-based autoregressive generative model, we verify that our efficient variable-sized shape tokenization is beneficial to downstream generation tasks. Experiments show our generated results are generally better than those of existing baselines regarding FID, KID, and CLIP scores.

4.2 Related Work

3D Generation. Recent 3D generation methods have achieved remarkable results by leveraging pre-trained large-scale 2D diffusion models [192]. Approaches like DreamFusion [179] and DreamGaussian [222] use 2D diffusion priors to optimize 3D representations, such as Neural Radiance Fields [149] and Gaussian Splats [90]. Subsequent works have improved performance with new loss functions and 3D representations [24, 89, 105, 111, 127, 130, 135, 144, 147, 217, 233, 249, 270]. However, these methods often require extensive iterative optimizations, making them impractical for real-world applications. To reduce inference time, feed-forward methods have been developed that synthesize multi-view consistent images of the same object followed by 3D reconstruction [66, 103, 116, 117, 122, 223, 231, 267, 281, 301].

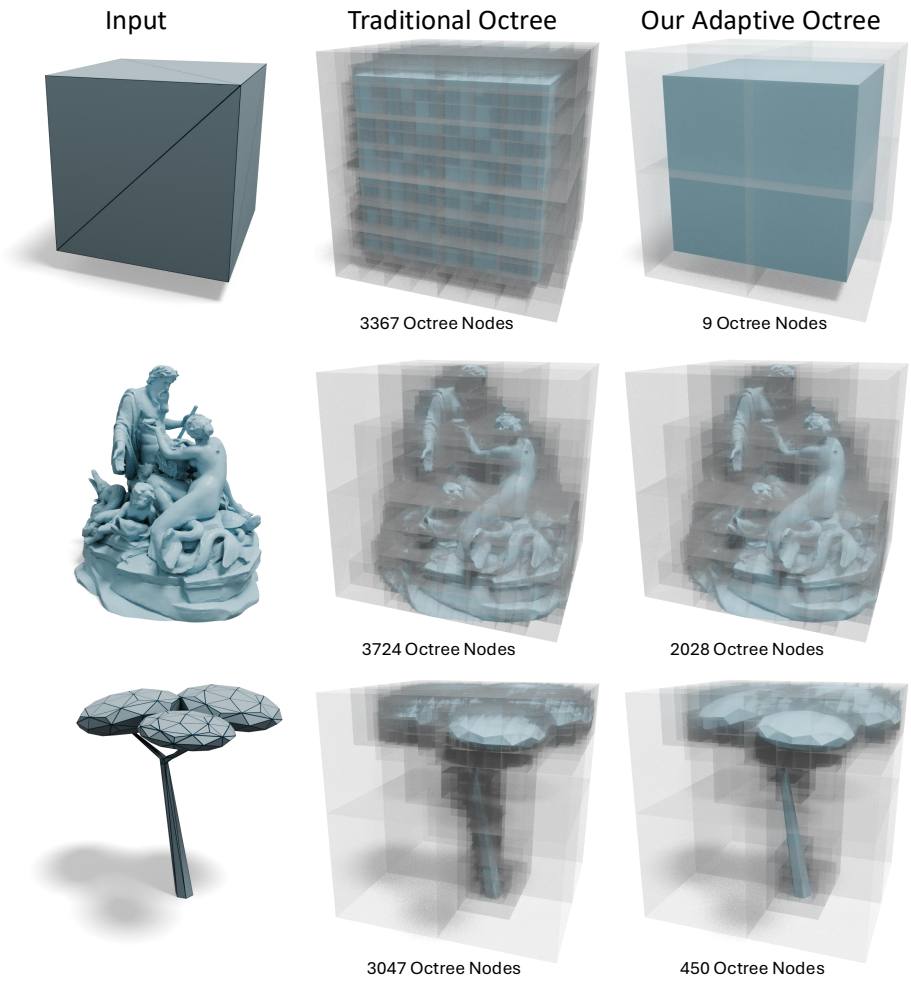


Figure 4.2: Traditional octree construction subdivides each octant based on whether the octant contains any mesh element. This construction always subdivides to the maximum depth (set to 6 in this example), leading to a similar amount of nodes for simple (top) and complex (middle) shapes. In contrast, our approach terminates subdivision when the local geometry is simple (e.g., a plane), leading to an adaptive octree that better reflects the shape complexity.

Nonetheless, approaches leveraging 2D diffusion priors alone without 3D-native supervision tend to suffer from challenges in modeling refined geometric structures and complex surfaces, especially for shapes of high concavity.

More recently, a wave of 3D-native generative models [106, 118, 188, 259, 261, 282, 288] has emerged, aiming to train directly on raw 3D assets rather than relying on 2D diffusion priors. These methods have achieved superior generation quality

compared to their predecessors thanks to the 3D-native architecture design. Another line of work explores auto-regressive methods for direct mesh generation with artist-like topology [28, 29, 58, 209, 224, 255]. Due to tokenization inefficiency and challenges in scaling up the context window, these methods are still struggling to model high-poly meshes with complex surfaces. In contrast, our work aims to explore more efficient tokenization schemes that encode shapes into compact yet expressive representations for 3D-native generation.

Compact 3D latent representations. Representing 3D shapes with compact latent representations has become increasingly popular in 3D generative modeling. One line of work, pioneered by 3DShape2VecSet [171], advocates encoding 3D shapes into latent vector sets that can be decoded into diverse geometry representations such as occupancy fields [106, 171, 257, 282, 287], signed distance fields [25, 288], and meshes [224]. These methods encode all shapes into a fixed-length vector, and do not adaptively adjust the representation budget based on shape complexity. Other work [206, 258] learns latent space from triplanes, but achieving high-fidelity triplane representations remains challenging, which limits their accuracy, especially for complex shapes.

An alternative direction focuses on structured 3D latent representations to better leverage the spatial hierarchies inherent in the underlying geometry. For instance, sparse voxel grids coupled with feature-rich latents or attributes, as proposed in XCube [188], MeshFormer [118], LTS3D [140] and Trellis [259], enables more efficient training for high-resolution shapes and scenes and better preservation of high-frequency geometric details. Meanwhile, OctFusion [261] proposes to represent a 3D shape as a volumetric octree with each leaf node encoded by latent features. Although these approaches offer adaptiveness in the latent representation similar to ours, their spatial structure is determined by volumetric occupancy rather than surface complexity.

Octree-based 3D representation. Octree [136, 137] is an efficient 3D data structure that recursively divides a 3D space into eight octants. It adapts to sparsity and minimizes storage and computation in empty regions, making it both memory- and computationally efficient. Compared to dense voxel grids, octrees significantly reduce memory usage while preserving fine geometric details in complex regions. Octree has been used in a wide range of geometric processing applications,

including point cloud compression [198], 3D texturing [8], multi-view scene reconstruction [218, 272], shape analysis [190, 242, 243], and shape generation [225, 261]. While similar adaptive octree [244] has been used for the shape classification and prediction tasks, our work is the first to explore octree representation in the context of 3D tokenization and autoregressive generation, which requires us to co-design the encoding, decoding, and generation with octree data structure. Compared to existing approaches [188, 259, 261] that use uniform tokenization schemes, our method adapts tokenization according to shape complexity.

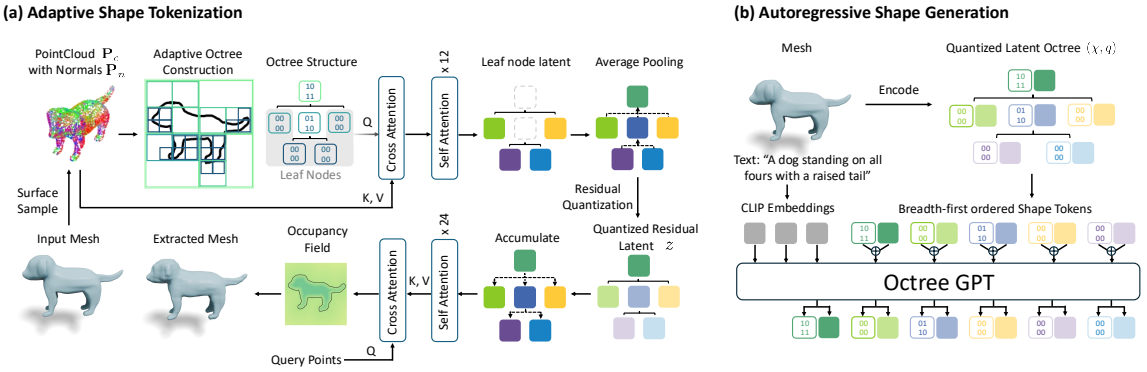


Figure 4.3: (a) **Adaptive Shape Tokenization.** Given an input mesh with surface point samples, we partition 3D space into a sparse octree that adapts to the local geometric complexity of the surface. We then use a Perceiver-based transformer [77] to encode the shape into a tree of latent codes, where a child node need encode only the (quantized) residual latent relative to its parent [100]. Latents can then be decoded into an occupancy field from which a mesh can be extracted. (b) **Autoregressive Shape Generation.** We define an autoregressive model for generating a tree of quantized shape tokens given a textual prompt, following a coarse-to-fine breadth-first search traversal. Similar to variable-length generation of text via end-of-sentence tokens, we make use of structural tokens to generate variable-size tree structures.

4.3 Method

Figure 4.3 illustrates our text-based shape generation framework. Our approach comprises two components: (1) a shape tokenization method (Octree-based Adaptive Tokenization, OAT) in Section 4.3.1 and Section 4.3.2 that efficiently compresses

3D shapes into a compact latent space, and (2) an autoregressive generative backbone model, OctreeGPT in Section 4.3.3, which operates on these variable-length shape tokens.

For each 3D shape, our approach begins by sampling a point cloud $\mathbf{P}_c \in \mathbb{R}^{N\times 3}$ from the surface, along with its surface normal vectors $\mathbf{P}_n \in \mathbb{R}^{N\times 3}$ following prior work [278]. We then employ our novel adaptive octree construction algorithm that partitions 3D space based on *local geometric complexity* to obtain an sparse octree structure. We then leverage the Perceiver-based transformer architecture [77] to encode the shape into an adaptive latent tree structure. The resulting variable-length latent representation can then be decoded into an occupancy field, from which a mesh can be extracted using marching cubes [129].

Unlike existing shape VAEs [106, 278, 282, 288] that learn fixed-size latent representations for every shape using Variational Autoencoders [94], we propose to encode shapes into variable-length latents based on their shape complexity. This adaptive tokenization approach results in a more compact latent space that only uses more latents by subdividing cells to finer resolution where the complexity of the shape is higher — thereby leading to better reconstruction quality and improved performance in downstream generative tasks.

4.3.1 Complexity-Driven Octree Construction

One of the core ingredients of our method is a sparse octree data structure which subdivides octants according to local geometry complexity, unlike existing methods subdividing cells based on occupancy.

An octree is a hierarchical spatial data structure that recursively subdivides 3D space into eight equal octants. Starting with a root node representing a bounding cube, each non-empty node can be further partitioned into eight child nodes, creating a tree-like structure $\mathcal{O} = \{\mathcal{V}, \mathcal{E}\}$. We use $\mathcal{V} = \{v_1, v_2, \dots\}$ to denote the cells in an octree hierarchy, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ defines parent-child relationships, where $(v_i, v_j) \in \mathcal{E}$ indicates that v_i is the parent of v_j. This representation is efficient to represent sparse 3D data, as it allocates higher resolution only to occupied regions.

In this paper, we consider the *sparse* octree by omitting empty child nodes,

i.e., each node can have 0 to 8 child nodes, with all nodes being non-empty. This structure can be compactly encoded by an 8-bit binary code $\chi : \mathcal{V} \to \{0,1\}^8$. For instance, $\chi(v) = (01001000)_2$ indicates that node v has two non-empty child nodes at its second and fifth slots. An octree structure can thus be *uniquely* represented as a sequence of 8-bit binary codes in breadth-first order, $[\chi(v_0), \chi(v_1), \cdots]$.

While octrees have previously been used to tokenize 3D shapes, earlier methods [188, 261] always subdivide up to the maximum depth unless empty. In contrast, we subdivide an octant only when the local geometry is "complex". Inspired by the literature in mesh simplification [49] and isosurfacing [83], we use the quadric error metric to measure shape complexity and guide octree subdivision. This approach optimizes representational capacity, allocating tokens where they provide the greatest benefit for shape fidelity.

Quadric error metric was first introduced to quantify local geometric complexity for mesh simplification tasks [50]. Given a plane in $\mathbb{R}^3$, let $\mathbf{p}$ denote a point on the plane with unit normal vector $\mathbf{n}$. The plane can be defined by all points $\mathbf{x} \in \mathbb{R}^3$ satisfying

$$\mathbf{n}^\top(\mathbf{x} - \mathbf{p}) = 0. \tag{4.1}$$

The quadric error measures the squared point-to-plane distance between any point $\mathbf{x}$ and this plane, computed as

$$E(\mathbf{x}) = \Big(\mathbf{n}^\top(\mathbf{x} - \mathbf{p})\Big)^2 \doteq [\mathbf{x}^\top, 1]\, \mathbf{Q}\, [\mathbf{x}^\top, 1]^\top, \tag{4.2}$$

where the quadric matrix $\mathbf{Q} \in \mathbb{R}^{4\times4}$ is defined as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{n}\mathbf{n}^\top & -\mathbf{n}\mathbf{n}^\top\mathbf{p} \\ (-\mathbf{n}\mathbf{n}^\top\mathbf{p})^\top & \mathbf{p}^\top\mathbf{n}\mathbf{n}^\top\mathbf{p} \end{bmatrix}. \tag{4.3}$$

As a key property, the cumulative error from a point $\mathbf{x}$ to *multiple* planes can be computed with a summed quadric,

$$E(\mathbf{x}) = \sum_i E_i(\mathbf{x}) = [\mathbf{x}^\top, 1] \left(\sum_i \mathbf{Q}_i\right) [\mathbf{x}^\top, 1]^\top. \tag{4.4}$$

We use the quadric error $E^* = \min_{\mathbf{x}} E(\mathbf{x})$ to measure local geometric complexity. As the energy is quadratic, the minimum E^* can be efficiently computed by solving a linear system, with details left in the appendix. Intuitively, when the planes form common intersections (e.g., an edge, a cone, or being flat), the optimal quadric error approaches zero, whereas complex regions usually yield higher quadric error values. This property makes quadric error metrics suitable for guiding adaptive geometric representations.

Specifically, for each octree cell $v \in \mathcal{V}$, we compute the cell quadric $\mathbf{Q}_v$ by summing up quadrics for all sampled points within v,

$$\mathbf{Q}_v = \sum_{\mathbf{p} \in \mathbf{P}_c(v)} \mathbf{Q}_\mathbf{p}, \tag{4.5}$$

where $\mathbf{P}_c(v) = \{\mathbf{p} \in \mathbf{P}_c \mid \mathbf{p} \text{ is contained in cell } v\}$ denotes the subset of points that lie within cell v, and $\mathbf{Q}_p$ is the quadric matrix for point $\mathbf{p}$ with its corresponding normal vector $\mathbf{n} \in \mathbf{P}_n$. We then calculate the average quadric error

$$E_v^* = \min_{\mathbf{x}} E_v(\mathbf{x}) = \frac{1}{|\mathbf{P}_c(v)|} \min_{\mathbf{x}} [\mathbf{x}^\top, 1] \mathbf{Q}_v [\mathbf{x}^\top, 1]^\top. \tag{4.6}$$

We recursively subdivide v into child cells only when both of these conditions are met: (1) the maximum depth L has not been reached, and (2) the quadric error exceeds our pre-defined threshold, $E_v^* > T$. In regions with complex geometry, cells are subdivided to the maximum depth L, while subdivision stops early in areas with simpler (i.e., planar) geometry.

4.3.2 Adaptive shape tokenization with OAT

Following prior work [106, 278, 282, 287, 288], we adopt a Perceiver-based variational autoencoder (VAE) [77, 94] to encode the shape into latents. Specifically, we compute:

$$\hat{\mathbf{P}} = \text{Concat}\left(\text{PE}(\mathbf{P}_c), \mathbf{P}_n\right), \tag{4.7}$$

$$\hat{\mathbf{O}} = \text{Concat}\left(\text{PE}(\mathcal{V}_{\text{leaf}}), \text{SE}(\mathcal{V}_{\text{leaf}})\right), \tag{4.8}$$

$$\phi(\mathcal{V}_{\text{leaf}}) = \text{SelfAttn}^{(i)}(\text{CrossAttn}(\hat{\mathbf{O}}, \hat{\mathbf{P}})),\ i = 1, \cdots, L_e,$$

where the encoder ϕ outputs a latent vector $\phi(v)$ for every leaf cell $v \in \mathcal{V}_{\text{leaf}}$, where $\phi : \mathcal{V} \to \mathbb{R}^d$. Here, PE denotes the positional encoding function [236], which operates on point coordinates and octree cell centers, while SE denotes the scale encoding function on the depth of octree cells. $\mathcal{V}_{\text{leaf}}$ comprises all the leaf cells within $\mathcal{V}$, and L_e refers to the number of Self Attention layers in the shape encoder.

Notably, the cross-attention operation is global, allowing each leaf cell to attend to all points in $\hat{\mathbf{P}}$ across the entire shape, rather than just points within its local cell. This global attention enables the model to capture long-range dependencies and contextual information beyond local neighborhoods. The subsequent self-attention layers further refine these representations by allowing leaf cells to exchange information.

Finally, we propagate latent vectors from leaf cells to their ancestors bottom-up. Each non-leaf node computes its latent vector by averaging those of its child nodes.

Multi-scale octree residual quantization. The variable length of the encoded latent motivates us to adopt an autoregressive model for downstream generation in Section 4.3.3. This approach requires us to learn a quantization bottleneck in the VAE. To achieve this, we propose an octree-based residual quantization strategy, enabling a coarse-to-fine token ordering using residual quantization [100, 229]. Specifically, we start quantization from the root node and only process the residual latent of every latent from its parent. We use a shared codebook and quantization function for all of the nodes using vqtorch [73]. We summarize our residual quantization algorithm in Algorithm 1.

Octree decoding. Given the multi-scale octree residual latent $z : \mathcal{V} \to \mathbb{R}^d$, we recover the full latent $\hat{\phi} : \mathcal{V} \to \mathbb{R}^d$ by adding the latent to every node from all its ancestors. Motivated by prior work [106, 278, 287], we use a similar perceiver-based transformer to decode the latent to an occupancy field. Specifically, given a query 3D point $\mathbf{x} \in \mathbb{R}^3$, the decoder predicts its occupancy value:

$$\hat{\mathbf{S}} = \text{Concat}\left(\hat{\phi}(\mathcal{V}), \text{PE}(\mathcal{V}), \text{SE}(\mathcal{V})\right), \tag{4.9}$$

$$\hat{\mathbf{S}} = \text{SelfAttn}^{(j)}(\hat{\mathbf{S}}), \qquad j = 1, 2, \cdots, L_d, \tag{4.10}$$

$$\hat{\sigma}(\mathbf{x}, \hat{\phi}, \mathcal{O}) = \text{CrossAttn}\left(\text{PE}(\mathbf{x}), \hat{\mathbf{S}}\right), \tag{4.11}$$

where L_d is the number of Self Attention layers in the shape decoder, and $\hat{\sigma}$ is the

Algorithm 1 Multi-scale octree residual quantization

Input: Octree $\mathcal{O} = \{\mathcal{V}, \mathcal{E}\}$, Latent $\phi : \mathcal{V} \to \mathbb{R}^d$.
Output: Multi-scale residual quantized latent $z : \mathcal{V} \to \mathbb{R}^d$, Quantized latent index $q : \mathcal{V} \to \mathbb{Z}$.

$z(v_0), q(v_0) = \text{Quantize}(\phi(v_0))$ ▷ v_0 is the root node.
$z_{acc}(v_0) = z(v_0)$ ▷ Initialize accumulated latent.
for $d = 1, \cdots, L-1$ **do** ▷ L is the max depth of $\mathcal{O}$.
  for $v \in \mathcal{V}_d$ **do** ▷ $\mathcal{V}_d$ is the set of nodes at level d.
    Find the parent v_{parent} of v according to $\mathcal{E}$.
    $z(v), q(v) = \text{Quantize}(\phi(v) - z_{acc}(v_{\text{parent}}))$.
    $z_{acc}(v) = z_{acc}(v_{\text{parent}}) + z(v)$. ▷ Update z_{acc}.
  end for
end for

predicted occupancy value at the query point. At inference time, we query the decoder using grid points and run marching cubes [129] to extract a mesh. During training, we sample query points using uniform and importance sampling near the mesh surface following prior work [106, 278, 287].We jointly optimize the networks and codebook via the following loss functions.

$$\mathcal{L}_{\text{VQ}} = \mathbb{E}_{v\in}||\text{sg}(\hat{\phi}(v)) - \phi(v)||^2 + ||\text{sg}(\phi(v)) - \hat{\phi}(v)||^2, \tag{4.12}$$

where $\text{sg}()$ is the stop-gradient operation. Additionally, we incorporate an occupancy reconstruction loss to ensure that the latent codes accurately reconstruct the input shape:

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{\mathbf{x}}\mathcal{L}_{\text{BCE}}\left(\sigma(\mathbf{x}), \hat{\sigma}(\mathbf{x}, \hat{\phi}, \mathcal{O})\right), \tag{4.13}$$

where $\mathcal{L}_{\text{BCE}}$ is the binary cross-entropy loss for shape reconstruction, and $\sigma(\mathbf{x}) \in \{0, 1\}$ is the ground truth occupancy value of the query point, indicating whether it is located inside the object. Our final loss function is:

$$\mathcal{L}_{\text{rec}} + \lambda_{\text{VQ}}\mathcal{L}_{\text{VQ}}, \tag{4.14}$$

where λ_{VQ} weights the vector quantization loss.

KL variant for continuous tokens. By replacing the quantization bottleneck with

a KL regularization [94], our proposed OAT can learn continuous shape latent instead, which provides a fair comparison with other continuous latent baselines in Section 4.4.1.

4.3.3 OctreeGPT: Autoregressive Shape Generation

Building on our adaptive tokenization framework, we develop OctreeGPT, an autoregressive model for generating 3D shapes conditioned on text descriptions. Unlike previous approaches that operate on fixed-size representations [106, 282, 288], OctreeGPT models the joint distribution of variable-length octree tokens while maintaining a hierarchical coarse-to-fine structure.

Shape Token Sequence. To enable autoregressive modeling, we first serialize the octree structure by traversing it in a breadth-first manner as mentioned in Section 4.3.2. For each node v, we include both its quantized index $q(v) \in \mathbb{Z}$ and a structural code $\chi(v) \in \{0,1\}^8$ that encodes the presence or absence of each potential child node. A latent octree can thus be *uniquely* represented by a variable-length sequence of tokens:

$$[t_0, t_1, \cdots, t_N],$$

where each token $t_i = (q(v_i), \chi(v_i)), \forall i \in \mathbb{N}$.

We train an autoregressive model that predicts the next token in the sequence,

$$P(t_0, t_1, \cdots, t_N|\theta) = \prod_{i=1}^{N} P(t_i|t_0, \cdots, t_{i-1}, \theta), \tag{4.15}$$

where θ is our learned OctreeGPT model.

Model Architecture. Our architecture builds upon decoder-only transformers similar to GPT-2 [45, 182]. Specifically, we compute the embedding for each shape token t_i as:

$$\text{Embed}(t_i) = \text{Embed}_q(q(v_i)) + \text{Embed}_\chi(\chi(v_i)) + \text{PE}_{\text{tree}}(v_i), \tag{4.16}$$

where $\chi(v_i)$ is interpreted as an 8-bit integer. The tree-structured positional encod-

ing $\mathrm{PE}_{\text{tree}}(v_i)$ captures both spatial and hierarchical information:

$$\mathrm{PE}_{\text{tree}}(v_i) = \mathrm{Embed}_x(x(v_i)) + \mathrm{Embed}_y(y(v_i)) \tag{4.17}$$

$$+\mathrm{Embed}_z(z(v_i)) + \mathrm{Embed}_d(d(v_i)), \tag{4.18}$$

where x, y, z are quantized coordinates of the cell center, and $d \in \{0, 1, \cdots, L-1\}$ is the depth of the octree node. This multi-dimensional positional encoding helps the model understand both spatial relationships and the hierarchical structure of the octree. Our model employs dual prediction heads for predicting quantized latent indices $\hat{q}$ and structural codes $\hat{\chi}$, allowing the model to jointly reason about geometry and tree structure. For text-conditioned generation, we prepend the sequence with 77 tokens derived from the input text's CLIP embedding [183].

Training and Inference. We train OctreeGPT using a combined loss function that balances the reconstruction of latent tokens and structural codes:

$$\mathcal{L}_{\text{GPT}} = \mathcal{L}_{\text{CE}}(q, \hat{q}) + \lambda_\chi \mathcal{L}_{\text{CE}}(\chi, \hat{\chi}), \tag{4.19}$$

where $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss for 2^8-way classification, and $\lambda\chi$ are balancing hyperparameters. During inference, we employ sampling with temperature τ to control the diversity and quality of generated shapes. We process the predicted structural code $\chi(v_i)$ on the fly to determine the octree topology, which dynamically establishes the final length of the token sequence. Further implementation details and hyperparameter settings are provided in the appendix.

4.4 Experiments

We evaluate our method on shape tokenization and generation. We perform qualitative and quantitative comparisons with existing baselines and conduct an ablation study on the significance of each major component.

Dataset. We use the Objaverse [37] dataset, which contains around 800K 3D models, as our training and test data. To ensure high-quality training and evaluation, we filter out low-quality meshes, such as those with point clouds, thin structures, or holes. This results in a curated dataset of around 207K objects for training and 22K

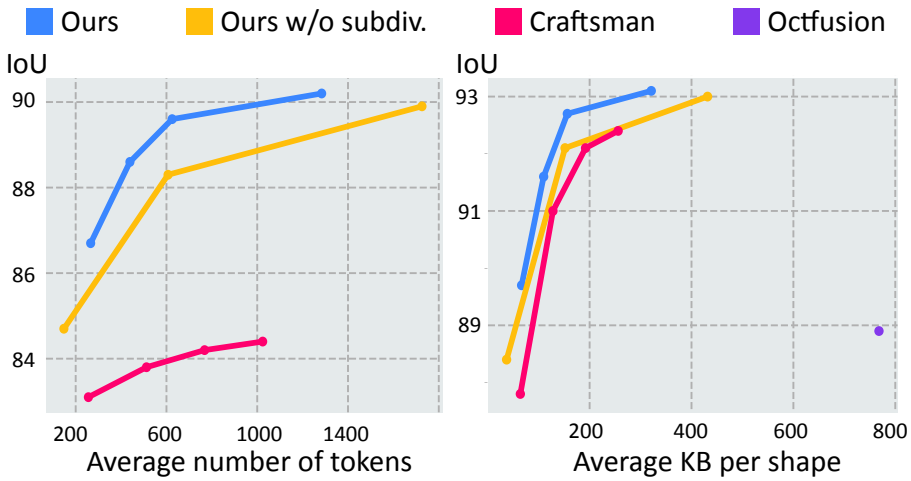


Figure 4.4: We plot reconstruction quality (IoU) against latent size in both discrete (left) and continuous (right) scenarios. We use KiloBytes (KB) for continuous latent representations for a fair comparison. Our method consistently outperforms baseline approaches at equivalent latent sizes and achieves comparable reconstruction quality with much smaller latent representations.

objects for testing.

For preprocessing, each mesh is normalized to a unit cube. For each mesh, we sample 1M points with their normals from the surface as the input point cloud. To generate ground-truth occupancy values, we uniformly sample 500K points within the unit volume and an additional 1M points near the mesh surface to capture fine details and obtain the occupancy based on visibility following prior work [282]. We then construct an adaptive octree for each shape based on the sampled point cloud using a pre-defined quadric error threshold T, which guides the subdivision process according to local geometric complexity. To enable text conditioning, we render nine views of each object under random rotations and use GPT-4o [2] to generate descriptive captions from these renderings.

4.4.1 Shape Reconstruction

We first assess the reconstruction fidelity of different latent representations.

Baselines. We compare OAT with latent vector sets from Craftsman3D [106]. For a fair comparison, we train both methods under identical conditions, using both quantization for discrete tokenization and KL regularization for continuous latent

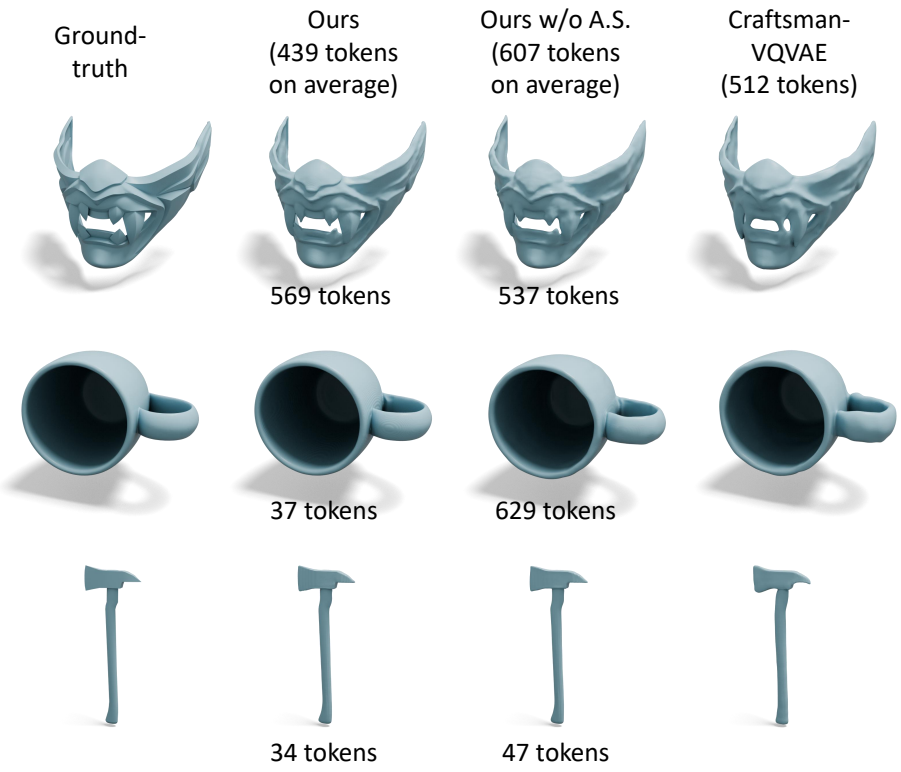


Figure 4.5: **Shape reconstruction with discrete latent.** We compare our full method against Craftsman-VQ [106] as well as an ablation without Adaptive Subdivision (A.S.). With comparable or lower token budget, our method generally outperforms the baseline regarding reconstruction fidelity. Meanwhile, without adaptive subdivision, the vanilla octree only allocates the token budget efficiently for objects of small volume (bottom) but wastes tokens on geometrically simple objects that occupy large space (middle).

space. Additionally, we evaluate against two other recent approaches, XCube [188] and Octfusion [261]. Due to computational resource constraints, we use publicly available pre-trained models for these two baselines rather than retraining them on our dataset. We exclude VAE models from Direct3D [257], CLAY [282], and LTS3D [140] as their implementations are not available.

Results. We evaluate shape reconstruction quality using volume Intersection over Union (IoU) and Chamfer Distance (CD) with 10K sampled surface points in Table 4.1 and Table 4.2. Note that XCube [188] outputs an Unsigned Distance Function (UDF), which cannot be evaluated with IoU metrics. Visual comparisons

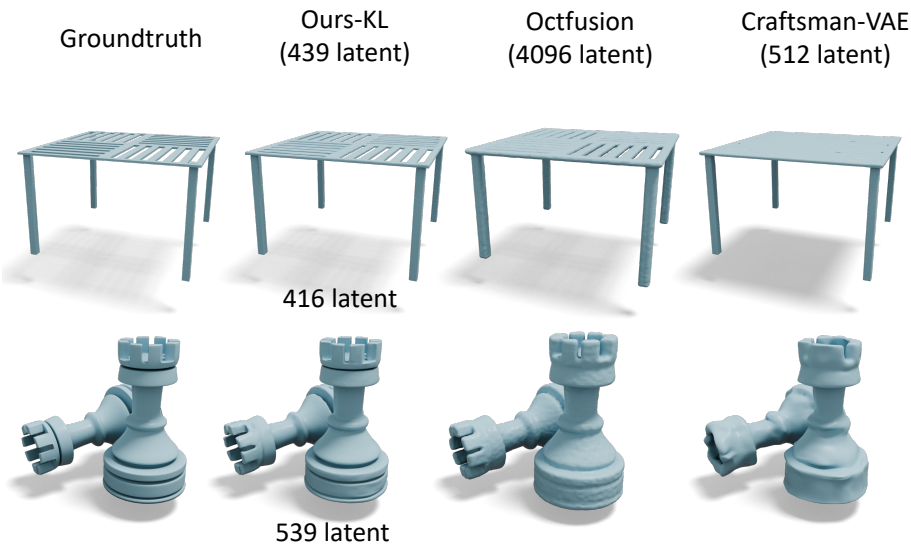


Figure 4.6: **Shape reconstruction with continuous latent.** We include the visual comparison between our continuous VAE (OAT-KL) and other baselines. In general, our reconstruction preserves more details using similar or smaller number of latent vectors.

in Figure 4.5 and Figure 4.6 demonstrate our approach outperforms all baselines.

Ablation Study. We ablate our proposed adaptive subdivision in Figure 4.5. Without quadric-error-based adaptive subdivision, the octree representation subdivides to the deepest level unless empty, wasting tokens on simple objects of large volumetric occupancy (middle row). Figure 4.4 shows reconstruction quality (IoU) versus latent size in both discrete and continuous scenarios, confirming our method achieves better quality at equivalent latent sizes and requires significantly smaller latent representations for comparable reconstruction quality. Figure 4.7 further shows a qualitative comparison between our method and the baseline in reconstruction quality with respect to the number of tokens used.

4.4.2 Shape Generation

This section evaluates our text-to-shape generation quality against multiple baselines. We train our OctreeGPT on top of OAT using 439 tokens on average, and for comparison, train a GPT model on Craftsman-VQ with 512 tokens. We include XCube [188]'s pre-trained Objaverse model as a native text-to-3D baseline. We also

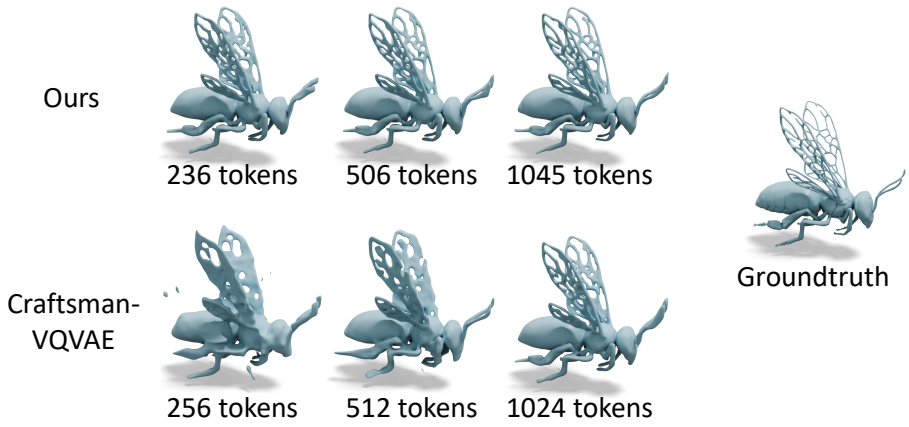


Figure 4.7: **Ablation study on token length.** With an increasing number of tokens, our method achieves better quality while consistently outperforming the baseline at a comparable token length.

compare against two image-to-3D methods, InstantMesh [262] and Craftsman [106], using FLUX.1 [96] to generate condition images from input text.

Results. We quantitatively evaluate generation quality in Table 4.3 by rendering generated shapes and computing Frechet Inception Distance (FID) [63, 176], and Kernel Inception Distance (KID) [10] against groundtruth renderings. We also report CLIP-score [183] to evaluate text-shape consistency, and average generation time to evaluate efficiency. In addition to quantitative measures, we also provide qualitative comparisons in Figure 4.8. Overall, thanks to a more compact and representative latent space, our OctreeGPT produces finer details with fewer artifacts compared to Craftsman-VQ with GPT, while also outperforming other 3D generation baselines in both geometry quality and prompt adherence, with a faster runtime.

4.5 Discussion

In this work, we propose an octree-based Adaptive Shape Tokenization, OAT, a framework that dynamically adjusts latent representations according to shape complexity. At its core, OAT constructs an adaptive octree structure guided by a quadric-error-based subdivision criterion, allocating more tokens to complicated parts and objects while saving on simpler ones. Extensive experiments show that

Method	Avg Token Cnt	IoU ↑	CD ($\times 10^{-3}$) ↓
Craftsman-VQ [106]	256	83.1	2.31
	512	83.8	1.94
	768	84.2	1.88
	1024	84.4	1.80
Ours (OAT) w/o A.S.	148	84.7	2.19
	607	88.3	1.85
	1726	89.9	1.37
Ours (OAT)	266	86.7	1.94
	439	88.6	1.78
	625	89.7	1.53
	1284	**90.2**	**1.27**

Table 4.1: **Quantitative analysis of shape reconstruction with discrete latent.** We compare our method against Craftsman-VQ [106] and ablation without Adaptive Subdivision (A.S.). With comparable token counts, our approach outperforms both baselines, showing the effectiveness of our proposed adaptive tokenization.

OAT reduces token counts by 50% compared to previous fixed-size approaches while maintaining comparable visual quality. Alternatively, with a similar number of tokens, OAT produces much higher-quality shapes. Building upon this tokenization, we develop an octree-based Autoregressive generative model, OctreeGPT that effectively leverages these variable-sized representations, outperforming existing baselines.

Limitations. Our framework only addresses geometric shape reconstruction and generation without incorporating texture information. We leave modeling both shape and texture properties jointly for future work.

Method	Avg Latent Len	IoU ↑	CD ($\times 10^{-3}$) ↓
Craftsman [106]	256	87.8	1.96
	512	91.0	1.83
	768	92.1	1.33
	1024	92.4	1.29
Octfusion† [261]	4096	88.9	1.87
XCube† [188]	4096	-	1.26
Ours (OAT-KL) w/o A.S.	148	88.4	1.89
	607	92.1	1.29
	1726	93.0	1.01
Ours (OAT-KL)	266	89.7	1.81
	439	91.6	1.29
	625	92.7	1.08
	1284	**93.1**	**0.97**

Table 4.2: **Quantitative analysis of shape reconstruction with continuous latent.** We replace the quantization with a KL regularization to learn continuous latent (OAT-KL) as mentioned in Section 4.3.2. Our method outperforms all the baselines with comparable or shorter latent code lengths. † indicates off-the-shelf models that are pre-trained on different data sources than ours.

Method	FID↓	KID↓ ($\times 10^{-3}$)	CLIP-score↑	Runtime↓ (secs)
Craftsman† [106]	65.18	6.42	0.27	54.8
InstantMesh† [153]	67.93	7.23	0.31	21.5
XCube [188]	132.56	9.83	0.23	32.3
Craftsman-VQ + GPT	85.10	7.49	0.26	15.4
Ours (OctreeGPT)	**56.88**	**5.79**	**0.34**	**11.3**

Table 4.3: **Quantitative analysis of shape generation.** We compare OctreeGPT with a GPT baseline trained on Craftsman-VQVAE (Section 4.4.1), text-to-3D model XCube [188], and image-to-3D methods InstantMesh [262] and Craftsman [106]. We compute FID [63], KID [10], and CLIP-score on the renderings of generated shapes, and report the average generation time. Our method outperforms all the baselines, showing higher quality and better consistency with the input text while achieving the fastest runtime due to our efficient tokenization.

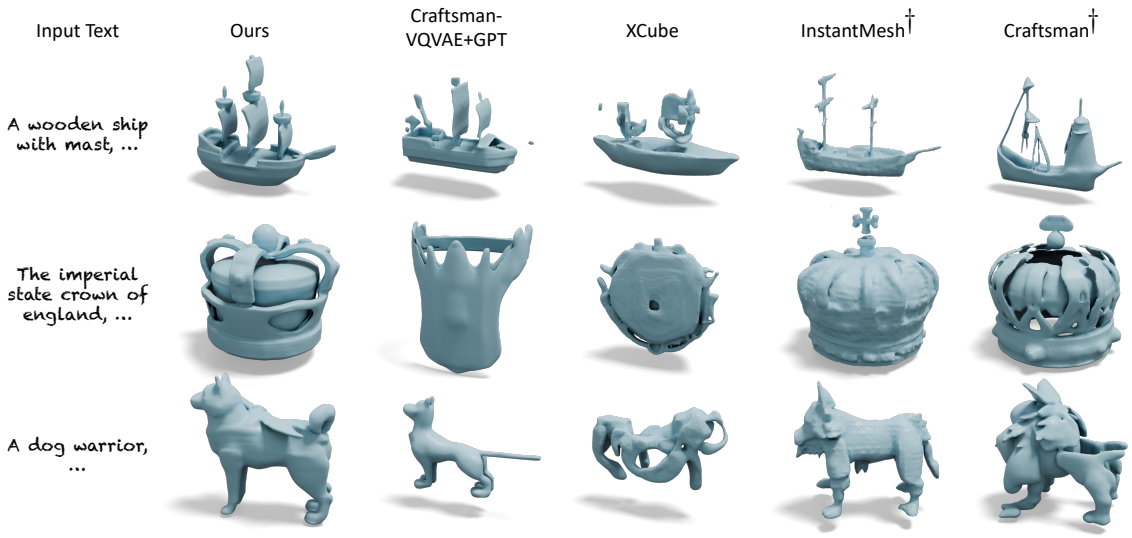


Figure 4.8: **Shape Generation Results.** We compare OctreeGPT with a GPT baseline trained on Craftsman-VQVAE (Section 4.4.1), text-to-3D model XCube [188], and image-to-3D methods InstantMesh [262] and Craftsman [106]. Our results have smoother surfaces, finer details, and fewer artifacts than baselines. For image-conditioned methods†, we use FLUX.1 [96] to generate condition images from input text.

Chapter 5

Fast Relightable Mesh Texturing with LightControlNet

Drawing inspiration from how professional artists create 3D content, we propose decomposing the challenge into two sequential stages: geometry generation (Chapter 4) and texture generation (This chapter). This decomposition allows each stage to be optimized with appropriate structural priors while maintaining the flexibility to create diverse, high-quality 3D assets from text descriptions.

Chapter 4 addressed the first stage by developing efficient octree-based representations for generating diverse 3D shapes from text prompts.

The following chapter tackles the challenge of generating high-fidelity relightable texture. Unlike simple flat texture, our approach generates physically-based materials that exhibit realistic behavior under various lighting environments. This capability is essential for downstream applications in games, films, and virtual environments, where generated assets must integrate convincingly with existing lighting setups and maintain visual consistency across different viewing conditions.

5.1 Introduction

Creating high-quality textures for 3D meshes is crucial across industries such as gaming, film, animation, AR/VR, and industrial design. Traditional mesh texturing tools are labor-intensive, and require extensive training in visual design. As the

demand for immersive 3D content continues to surge, there is a pressing need to streamline and automate the mesh texturing process (Figure 5.1).

In the past year, significant progress in text-to-image diffusion models [185, 192, 194] has created a paradigm shift in how artists create images. These models allow anyone who can describe an image in a text prompt to generate a corresponding picture. More recently, researchers have proposed techniques for leveraging such 2D diffusion models for automatically generating textures for an input 3D mesh based on a user-specified text prompt [23, 24, 144, 189]. But these methods suffer from three significant limitations that restrict their wide-spread adoption in commercial applications: (1) slow generation speed (taking tens of minutes per texture), (2) potential visual artifacts (e.g., seams, blurriness, lack of details), and (3) baked-in lighting causing visual inconsistency in new lighting environments (Figure 5.2). While some recent methods address one or two of these issues, none adequately address all three.

In this work, we propose an efficient approach for texturing an input 3D mesh based on a user-provided text prompt that disentangles the lighting from surface material/reflectance to enable relighting (Figure 5.1). Our method introduces **LightControlNet**, an illumination-aware text-to-image diffusion model based on the ControlNet [283] architecture, which allows specification of the desired lighting as a conditioning image for the diffusion model. Our text-to-texture pipeline uses LightControlNet to generate relightable textures in two stages. In stage 1, we use **multi-view visual prompting** in combination with the LightControlNet to produce visually consistent reference views of the 3D mesh for a small set of viewpoints. In stage 2, we perform a new **texture optimization** procedure that uses the reference views from stage 1 as guidance, and extends Score Distillation Sampling (SDS) [179] to work with LightControlNet. This allows us to increase the texture quality while disentangling the lighting from surface material/reflectance. We show that the guidance from the reference views allows our optimization to generate textures with over 10x speed-up than previous SDS-based relightable texture generation methods such as Fantasia3D [24]. Furthermore, our experiments show that the quality of our textures is generally better than those of existing baselines in terms of FID, KID, and user study.

Figure 5.1: We propose an efficient approach for texturing an input 3D mesh given a user-provided text prompt. Our generated texture can be relit properly in different lighting environments. The light probe shows the varied lighting environment. We suggest the readers check our video results of rotating lighting in our supplementary material.

5.2 Related Work

Text-to-Image generation. Recent years have seen significant advancements in text-to-image generation empowered by diffusion models [185, 192, 194]. Stable Diffusion [192], for example, trains a latent diffusion model (LDM) on the latent space rather than pixel space, delivering highly impressive results with affordable computational costs. Further extending the scope of text-based diffusion models, works such as GLIGEN [107], PITI [247], T2IAdapter [152], and ControlNet [283] incorporate spatial conditioning inputs (e.g., depth maps, normal maps, edge maps, etc.) to enable localized control over the composition of the result. Beyond their power in image generation, these 2D diffusion models, trained on large-scale text-

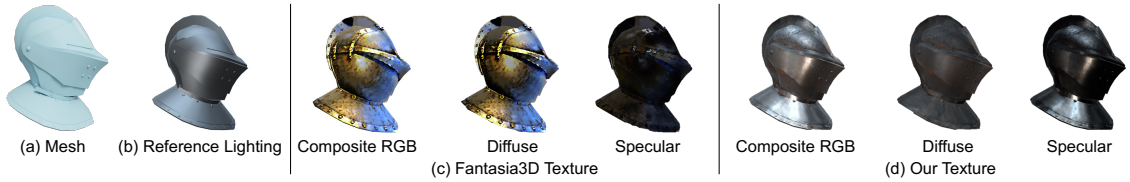


Figure 5.2: Given a 3D mesh of a helmet (a) and a lighting environment L, the reference rendering (b) depicts the "correct" highlights on the mesh due to L, by treating its surface reflectance as half-metal and half-smooth with a gray diffuse color. (c) The texture generated by the leading method Fantasia3D [24] is not properly relit as Fantasia3D bakes most of the lighting into the diffuse texture for the mesh and does not capture the bright highlights in the specular texture. (d) In contrast, our pipeline disentangles lighting from material, better capturing the diffuse and specular components of the metal helmet in this environment. Text prompt: "A medieval steel helmet."

image paired datasets, also contribute valuable priors to various other tasks such as image editing [62, 138], 3D generation [179, 184], and 3D editing [59, 95, 240, 300].

Text-to-3D synthesis. The success of text-to-image synthesis has sparked considerable interest in its 3D counterpart. Some approaches [104, 156, 206, 291] train a text-conditioned 3D generative model akin to 2D models, while others employ 2D priors from pre-trained diffusion models for optimization [24, 105, 111, 144, 179, 217, 241, 249] and multi-view synthesis [122, 204]. For instance, DreamFusion [179] and Score Jacobian Chaining [241] were the first to propose Score Distillation Sampling to optimize a 3D representation using 2D diffusion model gradients. Zero-1-to-3 [122] synthesizes novel views using a pose-conditioned 2D diffusion model. Yet, these methods often produce blurry, low-frequency textures that bake lighting into surface reflectance. Fantasia3D [24] can generate more realistic textures by incorporating physics-based materials. However, the resulting materials remain entangled with lighting, making it difficult to relight the textured object in a new lighting environment. In contrast, our method effectively disentangles the lighting and surface reflectance texture. Concurrent to our work, MATLABER [265] aims to recover material information in text-to-3D generation using a material autoencoder. Our method, however, differs in approach and improves efficiency.

3D texture generation. The area of 3D texture generation has evolved over time. Earlier models either directly took 3D representations as input to neural net-

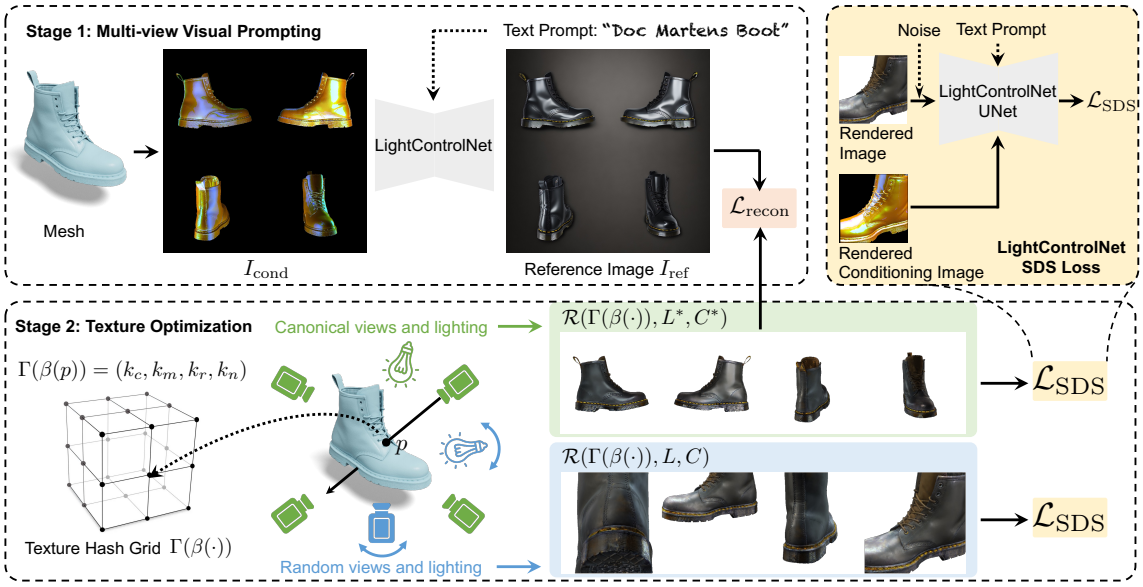


Figure 5.3: **Our Text-to-Texture pipeline.** Our method efficiently synthesizes relightable textures given an input 3D mesh and text prompt. In stage 1 (top left), we use ***multi-view visual prompting*** with our LightControlNet model to generate four visually consistent canonical views of the mesh under fixed lighting, concatenated into a reference image I_{ref}. In stage 2 we apply a new ***texture optimization*** procedure using I_{ref} as guidance along with a multi-resolution hash-grid representation of the texture $\Gamma(\beta(\cdot))$. For each optimization iteration, we render two batches of images using $\Gamma(\beta(\cdot))$ – one using the canonical views and lighting of I_{ref} to compute a reconstruction loss $\mathcal{L}_{\text{recon}}$ and the other using randomly sampled views and lighting to compute an SDS loss $\mathcal{L}_{\text{SDS}}$ based on LightControlNet.

works [11, 207, 274] or used them as templates [171, 178]. While some methods also use differentiable rendering to learn from 2D images [11, 60, 178, 274], the learned models often fail to generalize beyond the limited training categories.

Closest to our work are the recent works that use pre-trained 2D diffusion models and treat texture generation as a byproduct of text-to-3D generation. Examples include Latent-Paint [144], which uses Score Distillation Sampling in latent space, Text2tex [23], which leverages depth-based 2D ControlNet, and TEXTure [189], which exploits both previous methods. Nonetheless, similar to recent text-to-3D models, such methods produce textures with entangled lighting effects and suffer from slow generation. On the other hand, TANGO [30], generates material textures using a Spherical-Gaussian-based differentiable renderer, but struggles with

complex texture generation. A concurrent work, Paint3D [277], aims to generate lighting-less textures, yet it cannot produce material-based textures like ours.

Material generation. Bidirectional Reflection Distribution Function (BRDF) [161] is widely used for modeling surface materials in computer vision and graphics. Techniques for recovering material information from images often leverage neural networks to resolve the inherent ambiguities when applied to a limited range of view angles or unknown illuminations. However, these methods often require controlled setups [109] or curated datasets [9, 47, 252], and struggle with in-the-wild images. Meanwhile, material generation models like ControlMat [238], Matfuse [237], and Matfusion [197] use diffusion models for generating Spatially-Varying BRDF (SVBRDF) maps but limit themselves to 2D generation. In contrast, our method creates relightable materials for 3D meshes.

5.3 Preliminaries

Our text-to-texture pipeline builds on several techniques that have been recently introduced for text-to-image diffusion models. Here, we briefly describe these prior methods and then present our pipeline in Section 5.4.

ControlNet. ControlNet [283] is an architecture designed to add spatially localized compositional controls to a text-to-image diffusion model, such as Stable Diffusion [192], in the form of conditioning imagery (e.g., Canny edges [14], OpenPose keypoints [15], depth images, etc.). In our work, where we take a 3D mesh as input, the conditioning image $I_{\text{cond}}(C)$ is a rendering of the mesh from a given camera viewpoint C. Then, given text prompt y,

$$I_{\text{out}} = \text{ControlNet}(I_{\text{cond}}(C), y),$$

where the output image I_{out} is conditioned on y and I_{cond}. ControlNet introduces a parameter s that sets the strength of the conditioning image. When $s = 0$, the ControlNet simply produces an image using the underlying Stable Diffusion model, and when $s = 1$, the conditioning is strongly applied.

Score Distillation Sampling (SDS). DreamFusion [179] optimizes a 3D scene representation conditioned on text prompts using a pre-trained 2D text-to-image

diffusion model. The scene is represented as a NeRF [6, 149] parametrization θ. A differentiable renderer $\mathcal{R}$ applied to θ with a randomly sampled camera view C then generates a 2D image $x = \mathcal{R}(\theta, C)$. A small amount of noise $\epsilon \sim \mathbb{N}(0, 1)$ is then added to x to obtain a noisy image x_t. DreamFusion leverages a diffusion model ϕ (Imagen [194]) to provide a score function $\hat{\epsilon}_\phi(x_t; y, t)$, which predicts the sampled noise ϵ given the noisy image x_t, text prompt y, and noise level t. This score function can update the scene parameters θ, using the gradient calculated by Score Distillation Sampling (SDS):

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, x) = \mathbb{E}_{t,\epsilon}\left[w(t)(\hat{\epsilon}_\phi(x_t; y, t) - \epsilon)\frac{\partial x}{\partial \theta}\right],$$

where $w(t)$ is a weighting function. During each iteration, to calculate the SDS loss, we randomly choose a camera view C, render the NeRF θ to form an image x, add noise ϵ to it, and predict the noise using the diffusion model ϕ. We run the optimization for 5,000 to 10,000 iterations.

In our work, we introduce an illumination-aware SDS loss to optimize surface texture on a 3D mesh to suppress inconsistency artifacts and simultaneously separate lighting from the surface reflectance.

5.4 Method

Our text-to-texture pipeline operates in two main stages to generate a relightable texture for an input 3D mesh with a corresponding text prompt (Figure 5.3). In Stage 1, we use a **multi-view visual prompting** approach to obtain visually consistent views of the object from a small set of viewpoints, using a 2D ControlNet. Simply backprojecting these sparse views onto the 3D mesh could produce patches of high-quality texture, but would also generate visible seams and other visual artifacts where the views do not fully match. The resulting texture would also have lighting baked-in, making it difficult to relight the textured mesh in a new lighting environment. Therefore, in Stage 2, we apply a **texture optimization** that uses a ControlNet in combination with Score Distillation Sampling (SDS) [179] to mitigate such artifacts and separate lighting from the surface material properties/reflectance. In both stages, we introduce a new illumination-aware ControlNet

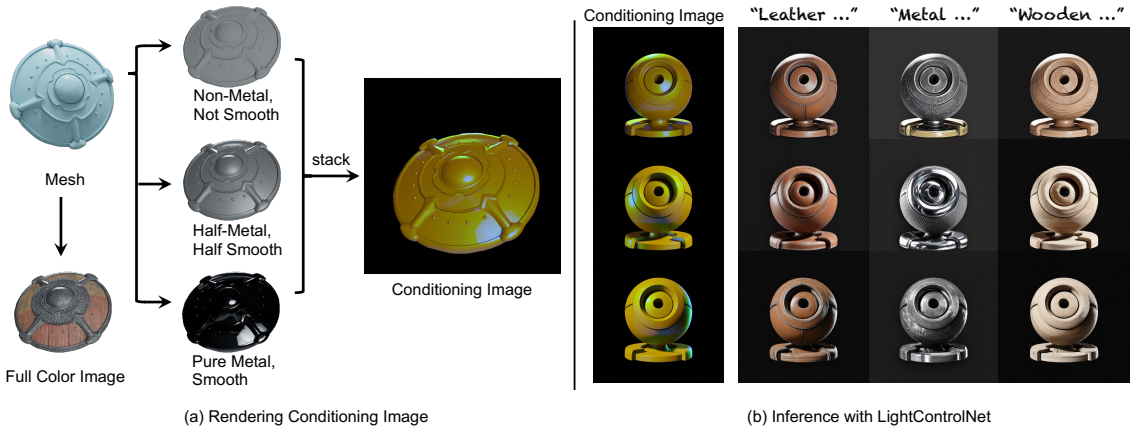


Figure 5.4: (a) LightControlNet requires a conditioning image that specifies desired lighting L for a view C of a 3D mesh. To form the conditioning image, we first render the mesh with the desired L and C using three different materials: (1) non-metal, not smooth, (2) half-metal, half-smooth, and (3) pure metal, smooth, and then combine the renderings into a single three-channel image. (b) LightControlNet is a diffusion model that is conditional on such light conditioning images as well as text prompts.

that allows us to specify the desired lighting as a conditioning image for an underlying text-to-image diffusion model. We call this model **LightControlNet** and describe how it works in Section 5.4.1. We then detail each stage in Section 5.4.2 and Section 5.4.3, respectively.

5.4.1 LightControlNet

LightControlNet adapts the ControlNet architecture to enable control over the lighting in the generated image. More specifically, we create a conditioning image for a 3D mesh by rendering it using three pre-defined materials and under known lighting conditions (Figure 5.4). These renderings encapsulate information about the desired shape and lighting for the object, and we stack them into a three-channel conditioning image. We have found that setting the pre-defined materials to (1) non-metal, not smooth; (2) half-metal, half-smooth; and (3) pure metal, extremely smooth, respectively, works well in practice. The specific material parameters are in the appendix.

To train our LightControlNet, we use 40K objects from the Objaverse dataset

[37]. Each object is rendered from 12 views using a randomly sampled camera C and lighting L sampled from 6 environment maps sourced from the Internet. L is also subject to random rotation and intensity scaling. For each resulting (L, C) pair, we render the conditioning image using the pre-defined materials, as well as the full-color rendering of the object using its original materials and textures. We use the resulting 480K pairs of (conditioning images, full-color rendering) to train LightControlNet using the approach of Zhang et al. [283].

Once LightControlNet is trained, we can specify the desired view and lighting for any 3D mesh. We first render the conditioning image with the desired view and lighting and then pass it along with a text prompt into LightControlNet, to obtain high-quality images. These images are spatially aligned to the desired view, lit with the desired lighting, and contain detailed textures (Figure 5.4).

Distilling the encoder. We improve the efficiency of LightControlNet by distilling the image encoder in Stable Diffusion [192], the base diffusion model in the ControlNet architecture. The original Stable Diffusion image encoder consumes almost 50% of the forward and backward time of SDS calculation using the latent diffusion model, primarily in downsampling the input image. Metzer et al. [144] have found the image decoder from latent space to image space can be closely approximated by per-pixel matrix multiplication. Inspired by this, we distill the encoder by removing its attention modules and training it on the COCO dataset [112] to match the original output. This distilled encoder runs 5x faster than the original one, resulting in an approximately 2x acceleration of our text-to-texture pipeline without compromising output quality. An ablation study of our distilled encoder is detailed in Table 5.3, with additional implementation specifics in the appendix.

5.4.2 Stage 1: Multi-view Visual Prompting

In Stage 1, we leverage LightControlNet to synthesize high-quality 2D images for a sparse set of views of the 3D mesh. Specifically, we create conditioning images for four canonical views C^* around the equator of the 3D mesh using a fixed lighting environment map L^* sampled from a set of environment maps. One approach to generating the complete texture for the mesh would be to apply the LightControlNet independently with each such conditioning image, but using the

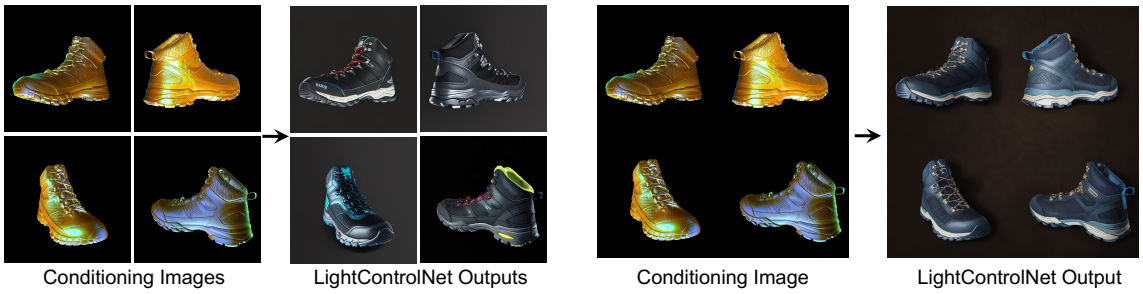


Figure 5.5: **Multi-view visual prompting.** (a) When we independently input four canonical conditioning images to LightControlNet, it generates four very different appearances and styles even with a fixed random seed. (b) When we concatenate the four images into a 2×2 grid and pass them as a single image into LightControlNet, it produces a far more consistent appearance and style. Text prompt: "A hiking boot".

same text prompt, and then backprojecting the four output images to the surface of the 3D mesh. In practice, however, applying the LightControlNet to each view independently produces inconsistent images of varying appearance and style, even when the text prompt and random seed remain fixed (Figure 5.5).

To mitigate this multi-view inconsistency issue, we take a multi-view visual prompting approach. We concatenate the conditioning images for the four canonical views into a single 2×2 grid and treat it as a single conditioning image. We observe that applying LightControlNet to all four views simultaneously, using this combined multi-view conditioning image, results in a far more consistent appearance and style across the views, compared to independent prompting (Figure 5.5). We suspect this property arises from the presence of similar training data samples – grid-organized sets depicting the same object – in Stable Diffusion's training set, which is also observed in concurrent works [253, 286]. Formally, we generate the conditioning image $I_{\text{cond}}(L^*, C^*)$ under a fixed canonical lighting condition L^* using four canonical viewpoints C^*. We then apply our LightControlNet with text prompt y to generate the corresponding reference image I_{ref}:

$$I_{\text{ref}} = \text{ControlNet}(I_{\text{cond}}(L^*, C^*), y).$$

5.4.3 Stage 2: Texture Optimization

In Stage 2, we could directly backproject the four reference views output in Stage 1 onto the 3D mesh using the camera matrix C associated with each view. While the resulting texture would contain some high-quality regions, it would also suffer from two problems (1) It would contain seams and visual artifacts due to remaining inconsistencies between overlapping views, occlusions in the views that leave parts of the mesh untextured, and loss of detail when applying the backprojection transformation and resampling the views. (2) In addition, as lighting is baked into the LightControlNet's RGB images, it would also be baked into the backprojected texture, making it difficult to relight the mesh.

To address both of these issues, we employ texture optimization using SDS loss. Specifically, we use a multi-resolution hash-grid [153] as our 3D scene representation, instead of NeRF as in the original DreamFusion formulation [179]. Given a 3D point $p \in \mathbb{R}^3$ on the mesh, our hash-grid produces a 32-dimensional multi-resolution feature. This feature is then fed to a 2-layer MLP Γ to obtain the texture material parameters for this point. Similar to Fantasia3D [24], these material parameters consist of metallicness $k_m \in \mathbb{R}$, roughness $k_r \in \mathbb{R}$, a bump vector $k_n \in \mathbb{R}^3$ and the base color $k_c \in \mathbb{R}^3$. Formally,

$$(k_c, k_m, k_r, k_n) = \Gamma(\beta(p)),$$

where β is the multi-resolution hash encoding function. Notably, this 3D hash-grid representation can be easily converted to 2D uv texture maps, which are more friendly to downstream applications. Given the mesh M, the texture $\Gamma(\beta(\cdot))$, a camera view C and lighting L we can use nvdiffrast [97], a differentiable renderer $\mathcal{R}$ to produce a 2D rendering of it, x, as

$$x = \mathcal{R}(M, \Gamma(\beta(\cdot)), L, C).$$

More details about the rendering equation are in the appendix. Since the mesh geometry is fixed, we omit M in the remainder of the paper.

Recall that the optimization approach of DreamFusion [179] randomly samples camera views C, generates an image for C using diffusion model ϕ, and supervises

the optimization using the SDS loss. We extend this optimization in two ways. First, we use four fixed reference images I_{ref} with their canonical views C^* and lighting L^* to guide the texture optimization through a reconstruction loss:

$$\mathcal{L}_{\text{recon}} = ||I_{\text{ref}} - \mathcal{R}(\Gamma(\beta(\cdot)), L^*, C^*)||_2 + \mathcal{L}_{\text{perceptual}}(I_{\text{ref}}, \mathcal{R}(\Gamma(\beta(\cdot)), L^*, C^*)),$$

where both L2 loss and perceptual loss [81] are used. For a non-canonical view C, we sample a random lighting L and use the SDS loss to supervise the optimization, but with our LightControlNet as the diffusion model ϕ_{LCN}, so

$$\nabla_{\Gamma,\beta}\mathcal{L}_{\text{SDS}}(\phi_{\text{LCN}}, x) = \mathbb{E}_{t,\epsilon}\left[w(t)(\hat{\epsilon}_{\phi_{\text{LCN}}}(x_t; y, t, I_{\text{cond}}(L, C)) - \epsilon)\frac{\partial x}{\partial \Gamma(\beta(\cdot))}\right],$$

where $x = \mathcal{R}(\Gamma(\beta(\cdot)), L, C)$ and $w(t)$ is the weight.

Finally, we employ a material smoothness regularizer on every iteration to enforce smooth base colors, using the approach of nvdiffrec [154]. For a surface point p with base color $k_c(p)$, the smoothness regularizer is defined as

$$\mathcal{L}_{\text{reg}} = \sum_{p \in S} |k_c(p) - k_c(p + \epsilon)|,$$

where S denotes the object surface and ϵ is a small random 3D perturbation. We use $\lambda_{\text{recon}} = 1000$ and $\lambda_{\text{reg}} = 10$ to reweight the loss $\mathcal{L}_{\text{recon}}$ and $\mathcal{L}_{\text{reg}}$.

Scheduling the optimization. We warm up the optimization by rendering the four canonical views and applying $\mathcal{L}_{\text{recon}}$ for 50 iterations. We then add in iterations using the $\mathcal{L}_{\text{SDS}}$ loss and optimize over randomly chosen camera views and randomly selected lighting from a pre-defined set of environmental lighting maps. Specifically we alternate iterations between using $\mathcal{L}_{\text{SDS}}$ and $\mathcal{L}_{\text{recon}}$. In addition, for a quarter of the SDS iterations, we use the canonical views rather than randomly selecting the views. This ensures that the resulting texture does not overfit to the reference images corresponding to the canonical views. The warm-up iterations capture the large-scale structure of our texture and allow us to use relatively small noise levels ($t \leq 0.1$) in the SDS optimization. We sample the noise following a linearly decreasing schedule [71] with $t_{\text{max}} = 0.1$ and $t_{\text{min}} = 0.02$. We also adjust

the conditioning strength s of our LightControlNet in the SDS loss linearly from 1 to 0 over these iterations so that LightControlNet is only lightly applied by the end of the optimization. We also experimented with a recent variant Variational Score Distillation [249], but did not observe notable improvement. We have experimentally found that we obtain high-quality textures after 400 total iterations of this optimization and this is far fewer iterations than other SDS-based texture generation techniques such as Fantasia3D [24] which requires 5000 iterations. More details are in the appendix.

Faster pipeline without relightability. Our two-stage pipeline is also compatible with off-the-shelf depth ControlNet and Stable Diffusion [192] as the backbone replacement of LightControlNet. Specifically, we can replace the LightControlNet in Stage 1 with a depth ControlNet that uses a depth rendering of the mesh as the conditioning image, and uses Stable Diffusion based SDS in Stage 2. In scenarios where texture relightability is not required, this variant offers an additional 2× speed-up (as shown in Table 5.1), since it eliminates the additional computation required by LightControlNet forward pass in the SDS optimization.

5.5 Experiments

In this section, we present comprehensive experiments to evaluate the efficacy of our proposed method for relightable, text-based mesh texturing. We perform both qualitative and quantitative comparisons with existing baselines, along with an ablation study on the significance of each of our major components.

Dataset. As illustrated in Figure 5.3, we employ Objaverse [37] to render paired data to train our LightControlNet. Objaverse consists of approximately 800k objects, of which we use the names and tags as their text descriptions. We filter out objects with low CLIP similarity [183] to their text descriptions and select around 40k as our training set. Each object is rendered from 12 views using randomly sampled cameras and lighting from a specific set of environmental lighting maps. To evaluate baselines and our method, we hold out 70 random meshes from Objaverse [37] as the test set. We additionally gather 22 mesh assets from 3D online games with 5 prompts each to assess our method, demonstrating its ability to generalize

Figure 5.6: **Sample results** from our method applied to Objaverse test meshes (top half) and 3D game assets (bottom half). To illustrate the efficacy of our relightable textures, for each textured mesh, we fix the environment lighting and render the mesh under different rotations. As shown above, our method is able to generate textures that are not only highly detailed, but also relightable with realistic lighting effects.

beyond Objaverse. Further details are in the appendix.

Baselines. We compare our approach with existing mesh texturing methods. Specifically, Latent-Paint [144] employs SDS loss in latent space for texture generation. Text2tex [23] progressively produces 2D views from chosen viewpoints, followed by an inverse projection to lift them to 3D. TEXTure [189] utilizes a similar lifting approach but supplements it with a swift SDS optimization post-lifting. Beyond

these texture generation methods, text-to-3D approaches serve as additional baselines, given that texture is a component of 3D generation. Notably, we choose Fantasia3D [24] as a baseline, the first to use a material-based representation for textures in text-to-3D processing.

Quantative evaluation. In Table 5.1, we compare our method with the baselines on the Objaverse [37] test set. For each method, we generate 16 views and evaluate Frechet Inception Distance (FID) [63, 176] and Kernel Inception Distance (KID) [10] compared with ground-truth rendered views. Two variations of our method are assessed. Both variants use our proposed two-stage pipeline, and the first employs a standard depth-guided ControlNet, while the second uses our proposed LightControlNet. Our method significantly outperforms the baselines in both quality and runtime.

(a) Close-up Comparison with Fantasia3D.
Left Prompt: "A medieval steel helmet" ; Right Prompt: "A leather horse saddle".

(b) Comparison with relightable and non-relightable baselines.
Top Prompt: "A hiking boot"; Bottom Prompt: "A leather horse saddle".

Figure 5.7: **Qualitative analysis.** (a) We compare our method with Fantasia3D [24] that also attempts to generate Physically Based Rendering (PBR) texture. However, unlike ours, their results often exhibit baked-in lighting, leading to artifacts when put into varied lighting environments. (b) We also compare our method with other baselines that can only generate non-relightable (RGB) texture. For non-relightable texture generation, we can replace our LightControlNet with depth ControlNet and generate RGB textures with a shorter runtime. More details are in Table 5.1.

	Objaverse test set		Game Asset		Runtime ↓
	FID ↓	KID ↓ ($\times 10^{-3}$)	FID ↓	KID ↓ ($\times 10^{-3}$)	(mins)
Latent-Paint [144]	73.65	7.26	204.43	9.25	10
Fantasia3D [24]	120.32	8.34	164.32	9.34	30
TEXTure [189]	71.64	5.43	103.49	5.64	6
Text2tex [23]	95.59	4.71	119.98	5.21	15
Ours (w/ depth)	**60.49**	3.96	85.92	3.87	**2**
Ours	62.67	**2.69**	**83.32**	**3.34**	4

Table 5.1: **Quantitative Evaluation.** We test our methods and baselines on 70 test objects from Objaverse [37] and 22 objects curated from 3D game assets. With depth ControlNet, our method yields superior results to all baselines while being three times as fast as the fastest baseline. Using LightControlNet (Ours) within our model improves the lighting disentanglement while maintaining comparable image quality.

Qualitative analysis. As shown in Figure 5.6, our method can generate highly-detailed textures that can be rendered properly with the environment lighting across a wide variety of meshes. We also visually compare our method and the baselines in Figure 5.7. Our method produces textures with higher visual fidelity than the baselines for both the relightable and non-relightable variants. In particular, when compared with Fantasia3D [24], a recent work that also aims to generate material-based texture, our results not only have superior visual quality, but also disentangle the lighting more successfully.

User study. To further evaluate the texture quality quantitatively, we conduct a user study comparing our results with each of the baselines on the Objaverse test set in Table 5.2. We ask 30 participants to evaluate (1) the realism of the results, (2) the consistency of the generated texture with the input text, and (3) the plausibility of the results when placed under varying lighting conditions. Each result is presented in the form of 360-degree rotation to display full texture details. The reference lighting is provided alongside when participants evaluate (3). Across all three aspects, participants consistently prefer our method over baselines.

Ablation study. We perform a thorough ablation analysis on different aspects of our method as seen in Table 5.3.When substituting our distilled encoder with

Preferred Percentage	Objaverse test set		
	Realistic	Consistent with text	Relightable
Ours v.s. Latent-Paint [144]	92.6%	74.5%	84.3%
Ours v.s. Fantasia3D [24]	81.9%	67.6%	74.3%
Ours v.s. TEXTure [189]	70.8%	57.3%	87.1%
Ours v.s. Text2tex [23]	75.4%	61.6%	88.6%

Table 5.2: **User study.** We conduct a user preference study to evaluate (1) result realism, (2) texture consistency with input text, and (3) plausibility under varied lighting. Participants consistently prefer our results over all baselines in these respects.

Objaverse test set	FID ↓	KID ($\times 10^{-3}$) ↓	Runtime ↓ (mins)
Ours (w/o dist. enc.)	**60.34**	2.84	8
Ours (w/o m.v.v.p)	74.23	3.54	19
Ours	62.67	**2.69**	**4**

Table 5.3: **Ablation study on algorithmic components.** We analyze the role of our distilled encoder (1st row) and multi-view visual prompting (2nd row). Replacing the distilled encoder with the original VQ-VAE encoder doubles the running time without a noticeable improvement. When removing the multi-view visual prompting for initial generation, the system requires 2,000 iterations (5x compared to our 400 iterations) to produce reasonable results, which produces slightly worse texture quality.

Material Basis			FID ↓	KID ($\times 10^{-3}$) ↓
non-metal, not smooth	half-metal, half-smooth	pure metal, smooth		
✓	✓	✓	**62.67**	**2.69**
	✓	✓	66.34	3.11
✓		✓	64.32	3.42
✓	✓		67.43	4.12
	✓		72.13	4.53

Table 5.4: **Ablation study on material bases.** We verify the impact of the material bases in rendering conditioning images. Omitting any one of these degrades quality.

Num. of canonical views	FID ↓	KID ($\times 10^{-3}$) ↓
2 views (front, back)	67.43	3.47
4 views (**Ours:** front, back, left, right)	**62.67**	**2.69**
6 views (front, back, left, right, top, bottom)	70.14	3.72

Table 5.5: **Ablation study on the number of canonical views.** We analyze the role of our canonical view selection in Section 5.4.2. Relying on only the left and right views provides insufficient supervision. Interestingly, adding top and bottom views leads to worse overall quality. This is likely due to the limitation of pre-trained 2D diffusion models in synthesizing top and bottom views well for a variety of objects. Furthermore, given the fixed resolution of the multi-view image, stacking more views would result in a lower resolution for each view, leading to a worse initialization for Stage 2.

the original VQ-VAE encoder, the performance is twice as slow, but the quality of results is not noticeably superior. On the other hand, without the multi-view visual prompting for the initial generation, the system requires 2000 iterations (a 5x slowdown compared to our 400 iterations) to produce reasonable results, while still leading to slightly worse texture quality.

In Section 5.4.1, we render a conditioning image using 3 pre-defined materials to encompass a broad range of feasible materials: (1) non-metal, not smooth (diffuse effect); (2) half-metal, half-smooth (mixed effect); (3) pure metal, smooth (specular effect). These material bases cover a large range of feasible materials. Table 5.4 shows omitting any one of these bases degrades quality.

As shown in Table 5.5, we also evaluate our selection of four canonical views in Section 5.4.2. Relying on only the left and right views provides insufficient supervision. Interestingly, incorporating top and bottom views degrades the performance. We hypothesize that the degradation is likely due to the limitation of 2D diffusion model backbones in generating top and bottom views reliably. Furthermore, stacking more views within a single image results in a decreased resolution for each view, given the fixed resolution of the multi-view image.

5.6 Discussion

We proposed an automated texturing technique based on user-provided prompts. Our method employs an illumination-aware 2D diffusion model (LightControlNet) and an improved optimization process based on the SDS loss. Our approach is substantially faster than previous methods while yielding high-fidelity textures with illumination disentangled from surface reflectance/albedo. We demonstrated the efficacy of our method through quantitative and qualitative evaluation on the Objaverse dataset and meshes curated from game assets.

Limitations. Our approach still poses a few limitations: (1) Baked-in lighting can still be found in certain cases, especially for meshes that are outside of the training data distribution of Objaverse; (2) The generated material maps are sometimes not fully disentangled and interpretable as metallicness, roughness, etc.; (3) Due to the inherent limitation of the 2D diffusion model backbones, the generated textures can fail to follow the text prompt in some cases.

Part III

Physical Asset Generation

Chapter 6

Generating Physically Stable and Buildable Brick Structures from Text

Chapters 4 and 5 established a complete pipeline for generating diverse, high-quality 3D assets from text descriptions by decomposing the problem into geometry and texture generation stages. This approach successfully produces visually compelling results that can be used in downstream virtual applications like games and film production.

However, a critical gap remains between generating visually plausible virtual 3D content and creating designs that are physically realizable in the real world. While our generated assets may look convincing on screen, they often violate fundamental physical principles, e.g., lacking structural stability, containing impossible geometries. This limitation becomes particularly important as 3D generation moves beyond entertainment and visualization toward applications in manufacturing, architecture, robotics, and other domains where physical realizability is essential.

The final chapter of this dissertation explores generating 3D content that is not only visually appealing but also structurally sound and buildable. Using LEGO brick construction as our testbed, the following chapter demonstrates how physical structural analysis can be integrated into modern generative models, producing designs that are simultaneously creative and physically stable.

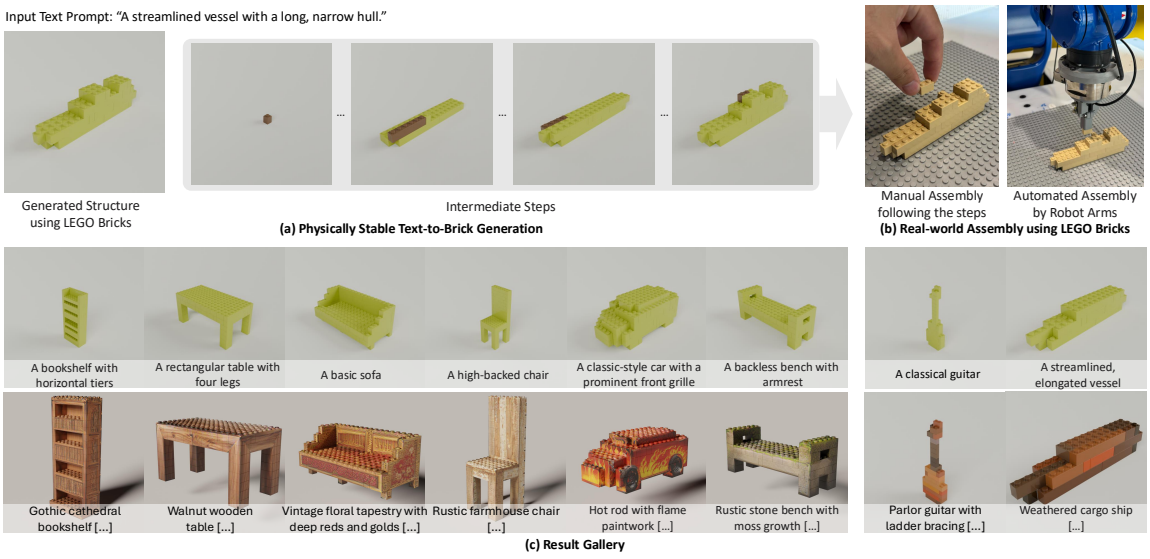


Figure 6.1: **Overview of BrickGPT.** (a) Our method generates physically stable interconnecting brick assembly structures from text descriptions through an end-to-end approach, showing intermediate brick-by-brick steps. (b) The generated designs are buildable both by hand and by automated robotic assembly. (c) We show example results with corresponding text prompts. Besides basic brick designs (top), our method can generate colored brick models (bottom right) and textured models (bottom left) with appearance descriptions. We highly recommend the reader to check our website for step-by-step videos.

6.1 Introduction

3D generative models have made remarkable progress, driven by advances in generative modeling [54, 211] and neural rendering [90, 149]. These models have enabled various applications in virtual reality, gaming, entertainment, and scientific computing. Several works have explored synthesizing 3D objects from text [179], adding texture to meshes [41, 189], and manipulating the shape and appearance of existing 3D objects and scenes [59, 124].

However, creating real-world objects with existing methods remains challenging. Most approaches focus on generating diverse 3D objects with high-fidelity geometry and appearance [66, 282], but these digital designs often cannot be physically realized due to two key challenges [133]. First, the objects may be difficult to assemble or fabricate using standard components. Second, the resulting structure

may be physically unstable even if assembly is possible. Without proper support, parts of the design could collapse, float, or remain disconnected.

In this work, we address the challenge of generating *physically realizable objects*. We study this problem in the context of designing structures made of interlocking toy bricks, such as LEGO® blocks. These are widely used in entertainment, education, artistic creation, and manufacturing prototyping. Additionally, they can serve as a reproducible research benchmark, as all standard components are readily available. Due to the significant effort required to design brick structures manually, recent studies have developed automated algorithms to streamline the process and generate compelling results. However, existing approaches primarily create structures from a given 3D object [132] or focus on a single object category [51, 52].

Our goal is to develop a method for generating brick assembly structures directly from freeform text prompts while ensuring physical stability and buildability. Specifically, we aim to train a generative model that produces designs that are:

- *Physically stable*: Built on a baseplate with strong structural integrity, without floating or collapsing bricks.
- *Buildable*: Compatible with standard interconnecting toy brick pieces and able to be assembled brick-by-brick by humans or robots.

In this work, we introduce BRICKGPT with the key insight of repurposing autoregressive large language models, originally trained for next-token prediction, for next-brick prediction. We formulate the problem of brick structure design as an autoregressive text generation task, where the next-brick dimension and placement are specified with a simple textual format.

To ensure generated structures are both *stable* and *buildable*, we enforce physics-aware assembly constraints during both training and inference. During training, we construct a large-scale dataset of physically stable brick structures paired with captions. During autoregressive inference, we enforce feasibility with an efficient validity check and physics-aware rollback to ensure that the final tokens adhere to physics laws and assembly constraints.

Our experiments show that the generated designs are stable, diverse, and visually appealing while adhering to input text prompts. Our method outperforms pre-trained LLMs with and without in-context learning, and previous approaches

based on 3D mesh generation. Finally, we explore applications such as text-driven brick texturing, as well as manual assembly and automated robotic assembly of our designs. Our dataset, code, and models are available at the project website: https://avalovelace1.github.io/BrickGPT/.

6.2 Related Work

Text-to-3D Generation. Text-to-3D generation has seen remarkable progress in recent years, driven by advances in neural rendering and generative models. Dreamfusion [179] and Score Jacobian Chaining [241] pioneer zero-shot text-to-3D generation by optimizing neural radiance fields [149] with pre-trained diffusion models [192]. Subsequent work has explored alternative 3D representations [24, 105, 111, 127, 144, 147, 217] and improved loss functions [89, 130, 135, 233, 249, 270]. Rather than relying on iterative optimization, a promising alternative direction trains generative models directly on 3D asset datasets, with various backbones including diffusion models [65, 104, 106, 156, 188, 206, 282, 288, 291], large reconstruction models [66, 103, 231, 267], U-Nets [118, 223], and autoregressive models [26, 28, 29, 58, 157, 208, 224, 255].

However, these existing methods cannot be directly applied to generating brick structures because they do not account for the unique physical constraints and assembly requirements of real-world designs [133]. Our work bridges this gap by introducing a method for generating physically stable and buildable brick structures directly from text prompts.

Autoregressive 3D Modeling. Recent research has successfully used autoregressive models to generate 3D meshes [26, 28, 29, 42, 58, 157, 208, 224, 255], often conditioned on input text or images. Most recently, LLaMA-Mesh [250] demonstrates that large language models (LLMs) can be fine-tuned to output 3D shapes in plain-text format, given a text prompt. However, most existing autoregressive methods focus on mesh generation. In contrast, we focus on generating brick structures from text prompts, leveraging LLMs' reasoning capabilities.

Brick Assembly and Design Generation. Creating brick structures given a reference 3D shape has been widely studied [92]. Existing works [164, 226, 290]

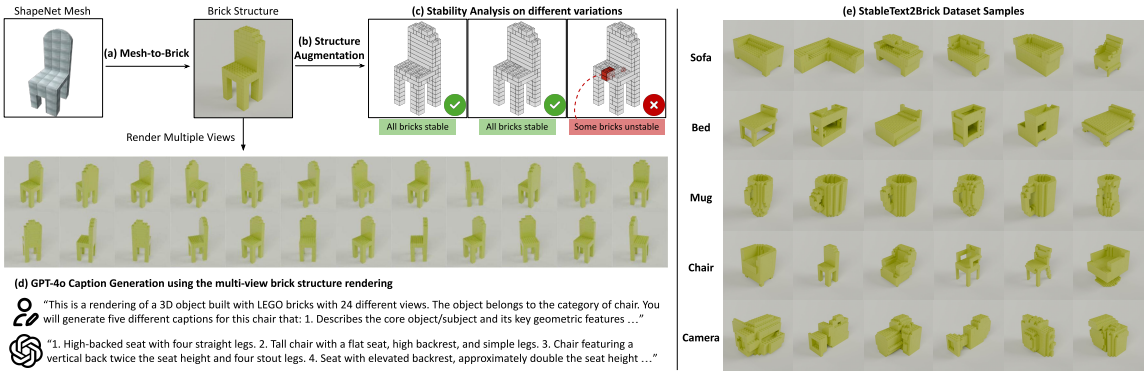


Figure 6.2: **StableText2Brick Dataset.** (a) From a ShapeNetCore [19] mesh, we generate a brick structure by voxelizing it onto a $20 \times 20 \times 20$ grid, then constructing its brick layout with a delete-and-rebuild algorithm. (b) We augment each shape with multiple structural variations by randomizing the brick layout while preserving the overall shape. (c) Stability analysis [119] is performed on each variation to filter out physically unstable designs. (d) To obtain captions for each shape, we render the brick structure from 24 different viewpoints and use GPT-4o [2] to generate detailed geometric descriptions. (e) Data samples from 5 categories in our StableText2Brick dataset.

formulate the generation as an optimization problem guided by hand-crafted heuristic rules. Such heuristics can include ensuring that all bricks are interconnected, minimizing the number of bricks, and maximizing the number of brick orientation alternations. Wang et al. [246] translate a visual manual into step-by-step brick assembly instructions. Luo et al. [132] leverage structural stability estimation to find weak structural parts and rearrange the local brick layout to generate physically stable designs. Kim et al. [93], Liu et al. [121] formulate a planning problem to fill the target 3D model sequentially. However, these methods only generate designs given an input 3D shape, assuming a valid brick structure exists, which is difficult to verify in practice.

Few works have explored learning-based techniques to generate toy brick designs. Thompson et al. [228] use a deep graph generative model in which the graph encodes brick connectivity. However, this method is limited to generating simple classes, such as walls and cuboids, using a single brick type. More recently, Ge et al. [52] use a diffusion model to predict a semantic volume, which is then translated into a high-quality micro building. Their method produces impressive results for a single category. Zhou et al. [289] and Ge et al. [51] generate compelling

figurine designs given an input portrait. They use machine learning to select from a pre-made set of components that best match an input photo. Although effective for faces, extending this selection-based approach to arbitrary objects is challenging. Zhou et al. [292] formulate an optimization problem to create a brick model from an input image. While their output is a 2D brick mosaic, we focus on 3D structures in this work. Goldberg et al. [53] query a vision-language model to generate diverse 3D assembly structures. However, they use regular building blocks instead of bricks with interlocking connections, and thus the structures have limited expressiveness.

Our goal is closest to that of [101]. This work has three steps: (1) generating an image using a text-to-image model, (2) converting the image into voxels, and (3) using heuristics to create a physical brick model without considering physical constraints. In contrast, our method performs the text-to-brick task without requiring intermediate image or voxel representations.

Physics-Aware Generation. Physics-aware 3D generation can be broadly categorized into two approaches: direct constraint enforcement and learned validation. Simple physical constraints, such as collision avoidance and contact requirements, can be incorporated directly through explicit penalty terms during optimization [56, 69, 125, 151, 239, 268, 275]. More complex physical properties, such as structural stability and dynamic behavior, typically require physics simulators [46, 146, 160, 264] or data-driven physics-aware assessment models [43, 145]. To our knowledge, our paper is the first attempt to incorporate physics-aware constraints into text-based brick assembly structure generation.

6.3 Dataset

Training a modern autoregressive model requires a large-scale dataset. Therefore, we introduce StableText2Brick, a new large-scale dataset that contains 47,000+ different toy brick assembly structures, covering 28,000+ unique 3D objects from 21 common object categories of the ShapeNetCore dataset [19]. We select categories featuring diverse and distinctive 3D objects while excluding those resembling cuboids. Each structure is paired with a group of text descriptions and a stability

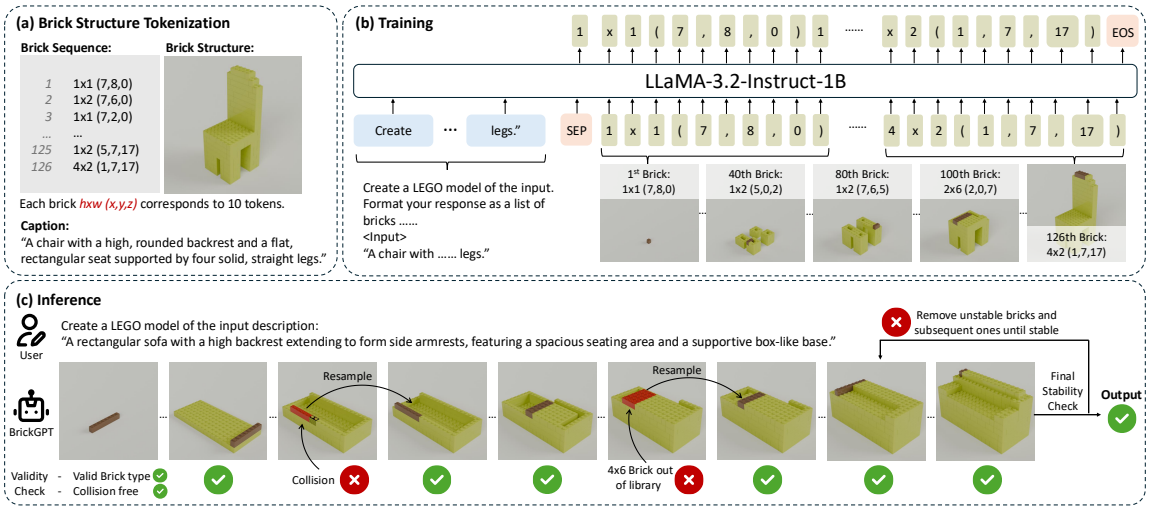


Figure 6.3: **Method.** (a) Our system tokenizes a brick structure into a sequence of text tokens, ordered in a raster-scan manner from bottom to top. (b) We create an instruction dataset pairing brick sequences with descriptions to fine-tune LLaMA-3.2-Instruct-1B. (c) At inference time, BRICKGPT generates brick structures incrementally by predicting one brick at a time given a text prompt. For each generated brick, we perform validity checks to ensure it is well-formatted, exists in our brick library, and does not collide with existing bricks. After completing the design, we verify its physical stability. If the structure is unstable, we roll back to a stable state by removing all unstable bricks and their subsequent bricks, and resume generation from that point.

score, which indicates its physical stability and buildability. Below, we describe the dataset construction, an overview of which is given in Figure 6.2.

Brick Representation. We consider brick structures built on a fixed baseplate. Each structure in StableText2Brick is represented as $B = [b_1, b_2, \ldots, b_N]$ with N bricks, and each element denotes a brick's state as $b_i = [h_i, w_i, x_i, y_i, z_i]$. Here, h_i and w_i indicate the brick length in the X and Y directions, respectively, and x_i, y_i, and z_i denote the position of the stud closest to the origin. The position has $x_i \in [0, 1, \ldots, H-1]$, $y_i \in [0, 1, \ldots, W-1]$, $z_i \in [0, 1, \ldots, D-1]$, where H, W, and D represent the dimensions of the discretized grid world.

Mesh-to-Brick. We construct the dataset by converting 3D shapes from ShapeNetCore [19] into brick structures as shown in Figure 6.2(a). Given a 3D mesh, we voxelize and downsample it into a $20 \times 20 \times 20$ grid world to ensure a consistent

scale, i.e., $H = W = D = 20$. The brick layout is generated by a delete-and-rebuild algorithm similar to [132]. To improve data quality and diversity, we introduce randomness and generate multiple different structures for the same 3D object, as illustrated in Figure 6.2(b). This increases the chance of obtaining a stable structure and more diverse layouts. We use eight commonly available standard bricks: 1×1, 1×2, 1×4, 1×6, 1×8, 2×2, 2×4, and 2×6.

Stability Score. We assess the physical stability of each structure, as illustrated in Figure 6.2(c), using the analysis method [119]. For a structure $B = [b_1, b_2, \ldots, b_N]$, the stability score $S \in \mathbb{R}^N$ assigns each brick b_i a value $s_i \in [0, 1]$ that quantifies the internal stress at its connections. Higher scores s_i indicate greater stability, while $s_i = 0$ indicates an unstable brick that will cause structural failure. Calculating the stability score requires solving a nonlinear program to determine the forces acting on each brick to achieve a static equilibrium that prevents structural collapse, as detailed in Section 6.4.2. For typically-sized (i.e., < 200 bricks) structures in Figure 6.2, stability analysis takes $\sim$0.35 seconds on average. A structure is stable if all bricks have stability scores greater than 0; we only include stable structures in the StableText2Brick dataset.

Caption Generation. To obtain captions for each shape, we render the brick structure from 24 different viewpoints and combine them into a single multi-view image. We then prompt GPT-4o [2] to produce five descriptions for these renderings with various levels of detail. Importantly, we ask GPT-4o to omit color information and focus only on geometry.

Figure 6.2(e) shows several data samples in StableText2Brick. The rich variations within each category and the comprehensive text-brick pairs make it possible to train large-scale generative models.

6.4 Method

Here, we introduce BRICKGPT, a method for generating physically stable interconnecting toy brick assembly structures from text prompts. Leveraging LLMs' ability to model sequences and understand text, we fine-tune a pre-trained LLM for the brick structure generation task (Section 6.4.1). To increase the stability and build-

ability of our designs, we use brick-by-brick rejection sampling and physics-aware rollback during inference (Section 6.4.2). Figure 6.3 illustrates an overview of our method.

6.4.1 Model Fine-tuning

Pre-trained LLMs excel at modeling sequences and understanding natural language, making them promising candidates for our task. We further fine-tune a pre-trained LLM on a custom instruction dataset containing text prompts and their corresponding brick structures from StableText2Brick.

Pre-trained Base Model. We use LLaMA-3.2-1B-Instruct [44] as our base model. This model is fine-tuned to give coherent answers to instruction prompts, making it suitable for text-based brick structure generation. As shown in Figure 6.5, the base model can generate brick structures through in-context learning, highlighting the promise of using pre-trained LLMs for our task. However, the generated structures fail to follow the input prompt, and they contain intersecting or disconnected bricks, making them physically unstable and unbuildable. To address these issues, we further fine-tune the pre-trained model using our StableText2Brick.

Instruction Fine-tuning Dataset. For each stable structure and its corresponding captions, we construct an instruction in the following format: "(user) Create a LEGO model of {caption}. (assistant) {brick-structure}."

To simplify training and reuse LLaMA's tokenizer, we represent brick structures in plain text. But what format should we use? The standard format LDraw [98] has two main drawbacks. First, it does not directly include brick dimensions, which are crucial for assessing the structure and validating brick placements. Second, it contains unnecessary information, such as brick orientation and scale. This information is redundant, as each axis-aligned brick has only two valid orientations.

Instead of using LDraw, we introduce a custom format to represent each brick structure. Each line of our format represents one brick as "$\{h\}\times\{w\}$ ($\{x\}$,$\{y\}$,$\{z\}$)", where $h \times w$ are brick dimensions and (x, y, z) are its coordinates. All bricks are 1-unit-tall, axis-aligned cuboids, and the order of h and w encodes the brick's orientation about the vertical axis. This format significantly reduces the number of

tokens required to represent a design, while including brick dimension information essential for 3D reasoning. Bricks are ordered in a raster-scan manner from bottom to top.

With our fine-tuned BRICKGPT model θ, we predict the bricks $b_1, b_2, ..., b_N$ in an autoregressive manner:

$$p(b_1, b_2, ..., b_N|\theta) = \prod_{i=1}^{N} p(b_i|b_1, ..., b_{i-1}, \theta). \tag{6.1}$$

6.4.2 Integrating Physical Stability

Although trained on physically stable data, our model sometimes generates designs that violate physics and assembly constraints. To address this issue, we further incorporate physical stability verification into autoregressive inference.

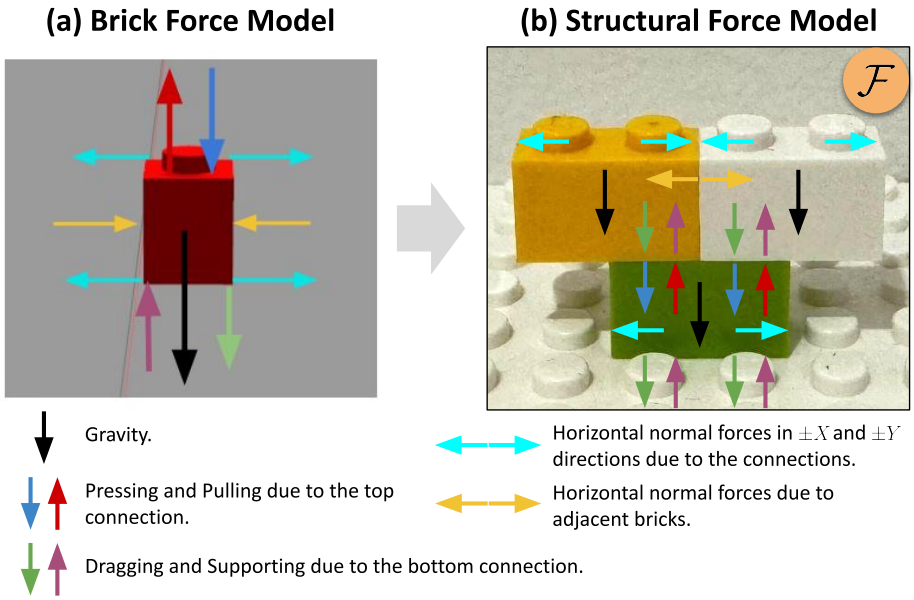


Figure 6.4: **Force Model.** (a) We consider all forces exerted on a single brick, including gravity (black), vertical forces with the top brick (red/blue) and bottom brick (green/purple), and horizontal (shear) forces due to knob connections (cyan), and adjacent bricks (yellow). (b) The structural force model $\mathcal{F}$ extends the individual force model to multiple bricks. Solving for static equilibrium in $\mathcal{F}$ determines each brick's stability score.

A brick structure is considered physically stable and buildable if it does not collapse when built on a baseplate. To this end, we assess physical structural stability using the stability analysis method [119]. We briefly overview this method below. Figure 6.4(a) illustrates all possible forces exerted on a single brick. We extend the single brick model and derive the structural force model $\mathcal{F}$, which consists of a set of candidate forces (e.g., pulling, pressing, supporting, dragging, normal, etc.), as shown in Figure 6.4(b). For a brick structure $B = [b_1, b_2, \ldots, b_N]$, each brick b_i has M_i candidate forces $F_i^j \in \mathcal{F}_i, j \in [1, M_i]$. A structure is stable if all bricks can reach static equilibrium, i.e.,

$$\sum_{j}^{M_i} F_i^j = 0, \qquad \sum_{j}^{M_i} \tau_i^j \doteq \sum_{j}^{M_i} L_i^j \times F_i^j = 0, \tag{6.2}$$

where L_i^j denotes the force lever corresponding to F_i^j. The stability analysis is formulated into a nonlinear program as

$$\underset{\mathcal{F}}{\operatorname{argmin}} \sum_{i}^{N} \left\{ \left| \sum_{j}^{M_i} F_i^j \right| + \left| \sum_{j}^{M_i} \tau_i^j \right| + \alpha \mathcal{D}_i^{\max} + \beta \sum \mathcal{D}_i \right\}, \tag{6.3}$$

subject to three constraints: 1) all force candidates in $\mathcal{F}$ should take non-negative values; 2) certain forces exerted on the same brick cannot coexist, e.g., the pulling (red arrow) and pressing (blue arrow), the dragging (green arrow) and supporting (purple arrow); 3) Newton's third law, e.g., at a given connection point, the supporting force on the upper brick should be equal to the pressing force on the bottom brick. $\mathcal{D}_i \subset \mathcal{F}_i$ is the set of candidate dragging forces (green arrow) on b_i. α and β are hyperparameter weights.

Solving the above nonlinear program in Equation 6.3 using Gurobi [57] finds a force distribution $\mathcal{F}$ that drives the structure to static equilibrium with the minimum required internal stress, suppressing the overall friction (i.e., $\sum \mathcal{D}_i$) as well as avoiding extreme values (i.e., $\mathcal{D}_i^{\max}$). From the force distribution $\mathcal{F}$, we

obtain the per-brick stability score as

$$s_i = \begin{cases} 0 & \sum_j^{M_i} F_i^j \neq 0 \\ & \vee \quad \sum_j^{M_i} \tau_i^j \neq 0 \\ & \vee \quad \mathcal{D}_i^{\max} > F_T, \\ \frac{F_T - \mathcal{D}_i^{\max}}{F_T} & \text{otherwise,} \end{cases} \tag{6.4}$$

where F_T is a measured constant friction capacity between brick connections. Higher scores s_i indicate greater stability, while $s_i = 0$ indicates an unstable brick that will cause structural failure: either $\mathcal{F}$ cannot reach static equilibrium ($\sum_j^{M_i} F_i^j \neq 0 \vee \sum_j^{M_i} \tau_i^j \neq 0$) or the required friction exceeds the friction capacity of the material ($\mathcal{D}_i^{\max} > F_T$). Due to the equality constraints imposed by Newton's third law, Equation 6.3 includes only the dragging forces and excludes pulling forces. For a physically stable structure, we need $s_i > 0, \forall i \in [1, N]$.

When to apply stability analysis? Our model generates structures sequentially, one brick at a time. A straightforward approach to ensuring physical stability is to apply stability analysis to each step and resample a brick that would cause a collapse. However, this step-by-step validation, though efficient per check, could be time-consuming due to the large number of checks required. More importantly, many structures are unstable when partially constructed, yet become stable when fully assembled. Adding a stability check after each brick generation could overly constrain the model exploration space. Instead, we propose brick-by-brick rejection sampling combined with physics-aware rollback to balance stability and diversity.

Brick-by-Brick Rejection Sampling. To improve inference speed and avoid overly constraining the model generation, we relax our constraints during inference. First, when the model generates a brick and its position, the brick should be well-formatted (e.g., available in the inventory) and not lie outside the workspace. Second, we ensure that newly added bricks do not collide with the existing structure. Formally, for each generated brick b_t, we have $\mathcal{V}_t \cap \mathcal{V}_i = \varnothing, \forall i \in [1, t-1]$, where $\mathcal{V}_i$ denotes the voxels occupied by b_i. These heuristics allow us to efficiently generate well-formatted brick structures without explicitly considering complex physical stability. To integrate these heuristics, we use rejection sampling: if a brick

Method	% valid	% stable	mean brick stability	min brick stability	CLIP	DINO
Pre-trained LLaMA (0-shot)	0.0%	0.0%	N/A	N/A	N/A	N/A
In-context learning (5-shot)	2.4%	1.2%	0.675	0.479	0.284	0.814
LLaMA-Mesh [250]	94.8%	50.8%	0.894	0.499	0.317	0.851
LGM [223]	**100%**	25.2%	0.942	0.231	0.300	0.851
XCube [188]	**100%**	75.2%	0.964	0.686	0.322	0.859
Hunyuan3D-2 [288]	**100%**	75.2%	0.973	0.704	0.324	0.868
Ours w/o rejection sampling or rollback	37.2%	12.8%	0.956	0.325	**0.329**	**0.888**
Ours w/o rollback	**100%**	24.0%	0.947	0.228	0.322	0.882
Ours (BRICKGPT)	**100%**	**98.8%**	**0.996**	**0.915**	0.324	0.880

Table 6.1: **Quantitative Analysis.** We evaluate our method against several baselines on validity (no out-of-library, out-of-bounds, or colliding bricks), stability, CLIP-based text similarity, and DINOv2-based image similarity. Stability, CLIP, and DINO are computed over valid structures only. For LLaMA-Mesh [250], validity requires a well-formed OBJ file. Our method outperforms all baselines as well as the ablated setups on validity and stability using our proposed rejection sampling and rollback, while maintaining high text similarity.

violates the heuristics, we resample a new brick from the model. Due to the relaxed constraints, most bricks are valid, and rejection sampling does not significantly affect inference time.

Physics-Aware Rollback. To ensure that the final design $B = [b_1, b_2, \ldots, b_N]$ is physically stable, we calculate the stability score S. If the resulting design is unstable, i.e., $s_i = 0, i \in \mathcal{I}$, we roll back the design to the state before the first unstable brick was generated, i.e., $B' = [b_1, b_2, \ldots, b_{\min \mathcal{I}-1}]$. Here, $\mathcal{I}$ is the set of the indices of all the unstable bricks. We repeat this process iteratively until we reach a stable structure B', and continue generation from the partial structure B'. Note that we can use the per-brick stability score to efficiently find the collapsing bricks and their corresponding indices in the sequence. We summarize our inference sampling in Algorithm 2.

6.4.3 Brick Texturing and Coloring

While we primarily focus on generating the *shape* of a brick structure, color and texture play a critical role in creative designs. Therefore, we propose a method that applies detailed UV textures or assigns uniform colors to individual bricks.

UV Texture Generation. Given a structure B and its corresponding mesh $\mathcal{M}$,

Algorithm 2 BRICKGPT inference algorithm.

Input: Text prompt c; Autoregressive model θ.
Output: Brick structure following the text prompt.

$B \leftarrow$ empty brick structure
loop ▷ Predict next brick w/ rejection sampling
  for $k = 1, \ldots,$ `max_rejections` **do**
    `context` $\leftarrow T \oplus B$.`to_text_format`()
    $b \leftarrow \theta$.`predict_tokens`(`context`) (Equation 6.1)
    if b is valid **then**
      break
    end if
  end for
  B.`add_brick`(b)
  if b contains EOF **then** ▷ Structure complete
    if B is stable or max rollbacks exceeded **then**
      return B
    end if
    while B is unstable **do** ▷ Rollback if unstable
      $\mathcal{I} \leftarrow$ indices of unstable bricks in B
      $i \leftarrow \min \mathcal{I}$ ▷ idx of 1st unstable brick
      $B \leftarrow [b_1, \ldots, b_{i-1}]$
    end while
  end if
end loop

we first identify the set of occluded bricks B_{occ} that have all six faces covered by adjacent bricks, and remove B_{occ} for efficiency. The remaining bricks $B_{\text{vis}} = B \setminus B_{\text{occ}}$ are merged into a single mesh $\mathcal{M}$ with cleaned overlapping vertices using *ImportLDraw* [230]. We generate a UV map $\text{UV}_{\mathcal{M}}$ by cube projection. The texture map I_{texture} is then generated using FlashTex [41], a fast text-based mesh texturing approach:

$$I_{\text{texture}} = \text{FlashTex}(\mathcal{M}, \text{UV}_{\mathcal{M}}, c), \tag{6.5}$$

where text prompt c describes the visual appearance. This texture can be applied through UV printing or stickers.

Uniform Brick Color Assignment. We can also assign each brick a uniform color from a standard color library [98]. Given a structure B, we convert it to a voxel

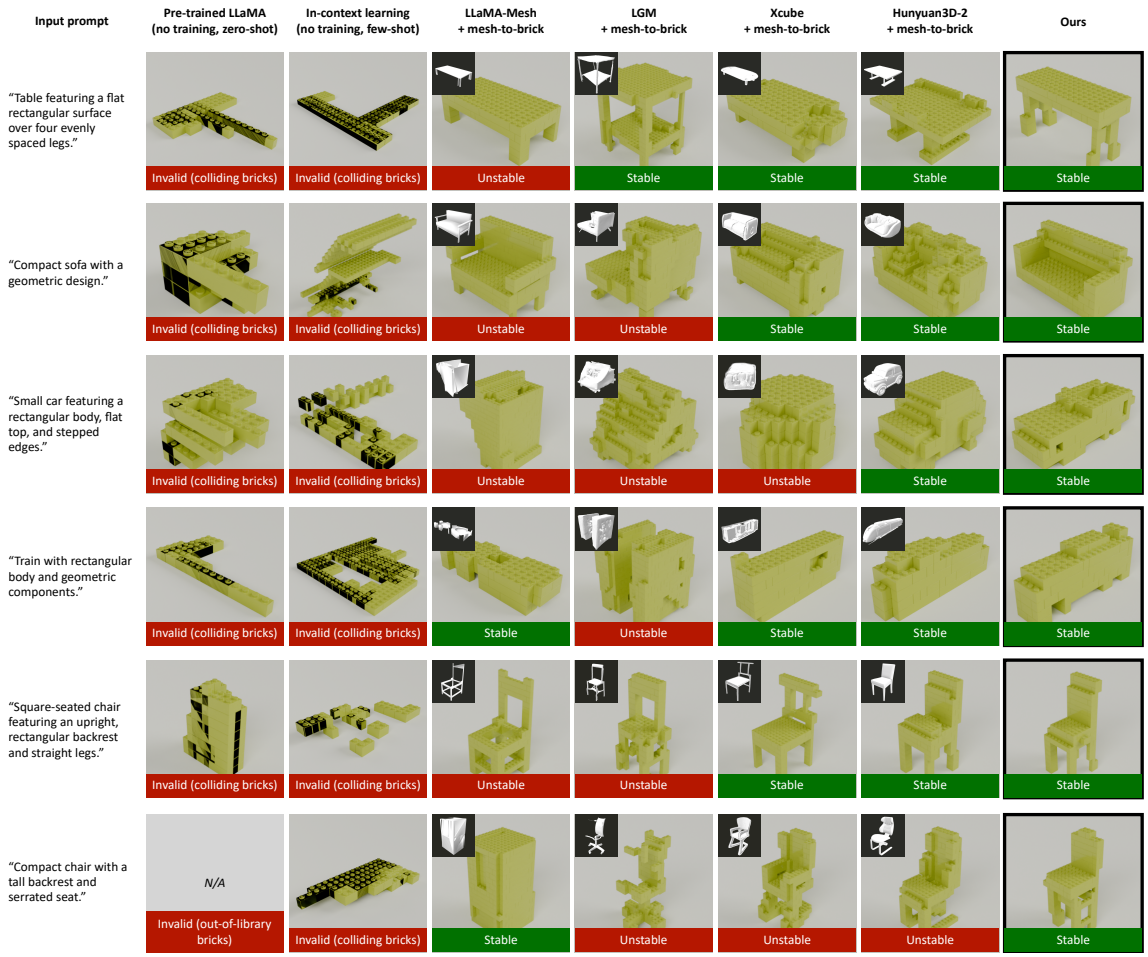


Figure 6.5: **Result gallery and baseline comparisons.** Our method generates high-quality, diverse, and novel brick structures aligned with the given text prompts. Black bricks are colliding. For LLaMA-Mesh [250], LGM [223], XCube [188], and Hunyuan3D-2 [288], an inset of the generated mesh is shown in the top-left corner.

grid $\mathcal{V}$ and then to a UV-unwrapped mesh $\mathcal{M}_{\mathcal{V}}$. For every voxel $v \in \mathcal{V}$, let $f_i^v, i = 1, \ldots, N_v$ be its visible faces where $0 \leq N_v \leq 6$. Each face f_i^v is split into two triangles and mapped to a UV region $\mathcal{S}_i^v$, creating a mesh $\mathcal{M}_{\mathcal{V}}$ with UV map $\mathrm{UV}_{\mathcal{V}}$. We apply FlashTex [41] to generate a texture I_{texture}:

$$I_{\text{texture}} = \text{FlashTex}(\mathcal{M}_{\mathcal{V}}, \mathrm{UV}_{\mathcal{V}}, c). \tag{6.6}$$

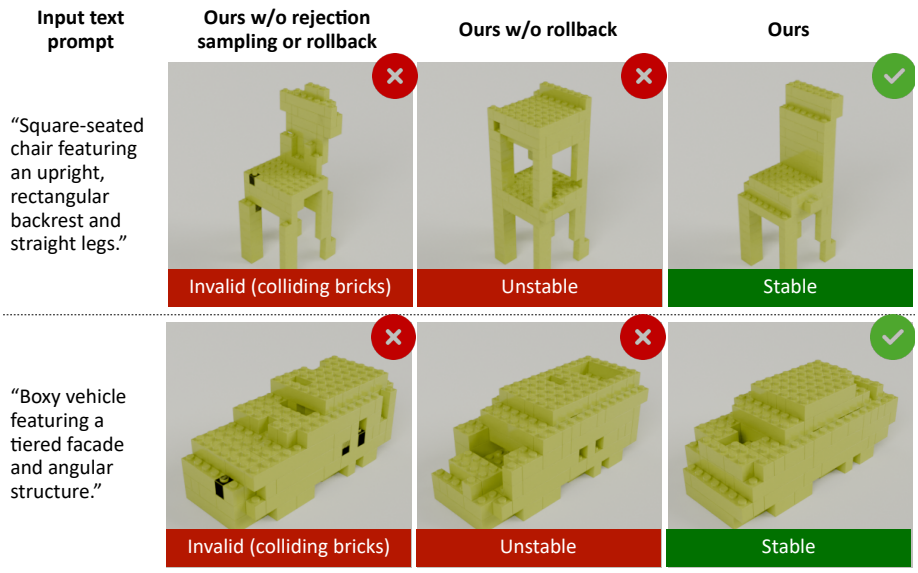


Figure 6.6: **Ablation study.** Brick-by-brick rejection sampling and physics-informed rollback help to ensure that the generated structure is both valid and stable. Black indicates colliding bricks.

The color of each voxel $\mathcal{C}(v) \in \mathbb{R}^3$ is computed as:

$$\mathcal{C}(v) = \frac{1}{N_v} \sum_{i=1}^{N_v} \mathcal{C}(f_i^v), \quad \forall v \in \mathcal{V}, \tag{6.7}$$

where $\mathcal{C}(f_i^v) = \frac{1}{|\mathcal{S}_i^v|} \sum_{(x,y)\in\mathcal{S}_i^v} I_{\text{texture}}(x,y)$ is the color of each visible face f_i^v, and $|\mathcal{S}_i^v|$ represents the number of pixels in region $\mathcal{S}_i^v$ in the UV map. For each brick b_t and its constituent voxels $\mathcal{V}_t$, we compute the brick color $\mathcal{C}(b_t) = \frac{1}{|\mathcal{V}_t|} \sum_{v\in\mathcal{V}_t} \mathcal{C}(v)$. Finally, we find the closest color in the color set. While UV texturing offers higher-fidelity details, uniform coloring allows us to use standard toy bricks.

6.5 Experiments

6.5.1 Implementation Details

Fine-tuning. Our fine-tuning dataset contains 240k distinct prompts and 47k+ distinct brick structures. We use 90% of the data for training and 10% for evaluation.

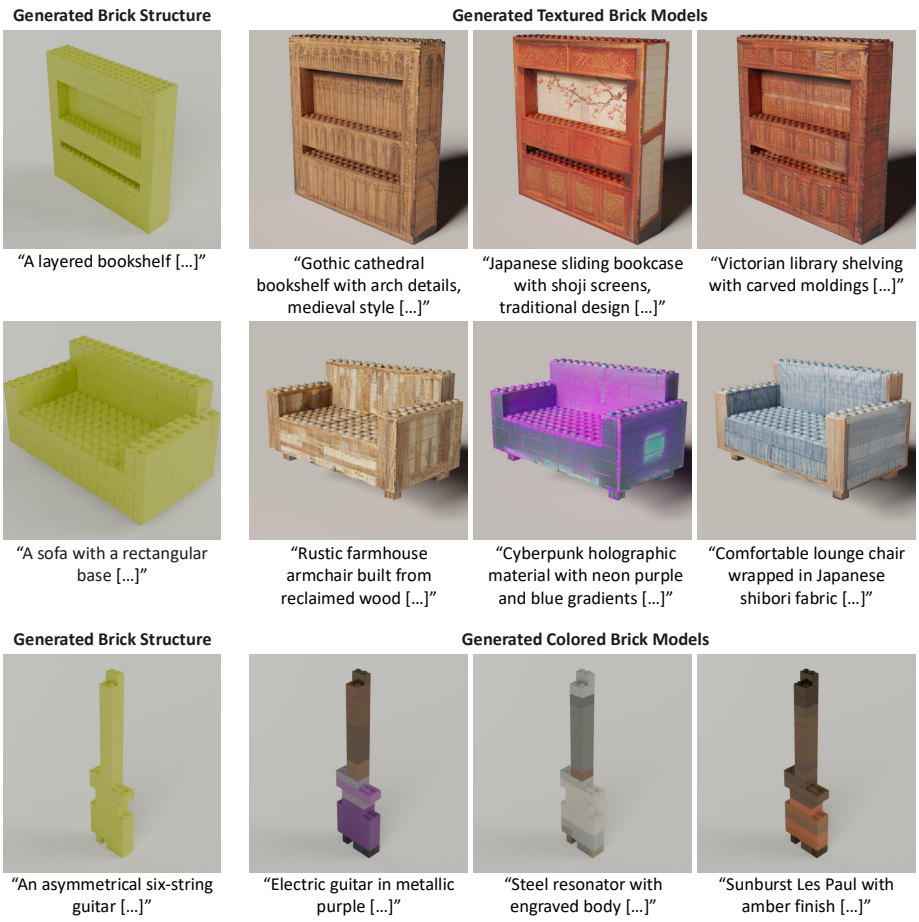


Figure 6.7: **Brick Texture and Color Generation.** Our method can generate diverse textured (top two rows) and colored (bottom) brick structures based on the same shape while using different appearance text prompts.

For efficiency, we include samples only up to 4096 tokens in length.

Inference. To evaluate our method, we generate one brick structure for each of 250 prompts randomly selected from the validation dataset. The nonlinear optimization in Equation 6.3 is solved using Gurobi [57]. We set $F_T = 0.98N$ with $\alpha = 10^{-3}$ and $\beta = 10^{-6}$. We allow up to 100 physics-aware rollbacks before accepting the brick structure. The median number of required rollbacks is 2, and the median time to generate one structure is 40.8 seconds.

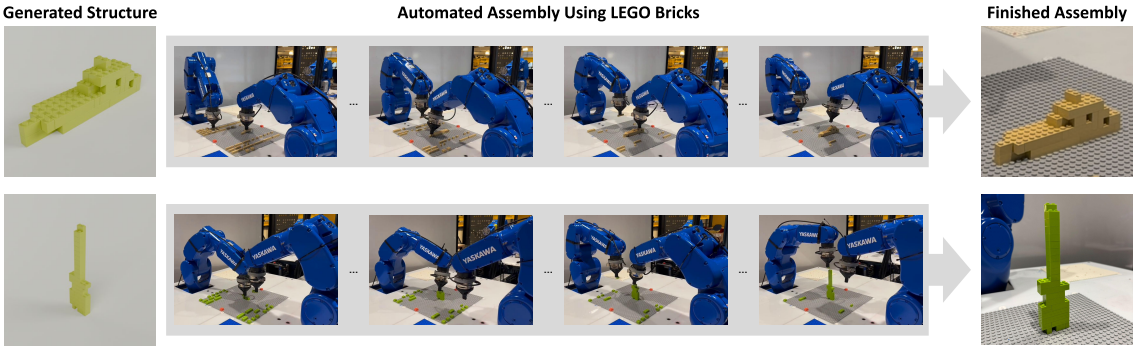


Figure 6.8: **Automated Assembly.** We demonstrate robotic assembly of generated structures using LEGO bricks.

6.5.2 Brick Structure Generation Results

Figure 6.5 shows a gallery of diverse, high-quality brick structures that closely follow the input prompts.

Baseline Comparisons. As baselines, we use LLaMA-Mesh [250], LGM [223], XCube [188], and Hunyuan3D-2 [288] to generate a mesh from each prompt, then convert the meshes to brick structures with our delete-and-rebuild algorithm. Additionally, we compare our method with pre-trained LLaMA models evaluated in both a zero-shot and few-shot manner. For few-shot evaluation, we provide the model with 5 examples of stable brick structures and their captions.

For each method, we compute the proportion of stable and valid structures among the generated designs. Additionally, for each valid structure, we compute its mean and minimum brick stability scores. To evaluate prompt alignment, we compute the CLIP score [183] between a rendered image of each valid structure and the text "A LEGO model of {prompt}". We also calculate the alignment between rendered images of the generated structure and the ground-truth structure for the same prompt, as measured by the cosine similarity between DINOv2 features [165]. As shown in Table 6.1, our method outperforms all baselines in these metrics. Figure 6.5 shows that our method generates brick structures of higher quality than the baselines.

Ablation Study. We demonstrate the importance of rejection sampling and physics-aware rollback. As seen in Figure 6.6, rejection sampling eliminates invalid bricks, such as those with collisions, while rollback helps to ensure the final assembly

structure is physically stable. The quantitative results in Table 6.1 show that our full method generates a higher proportion of valid and stable brick structures, while closely following the text prompts.

6.5.3 Extensions and Applications

Robotic Assembly of Generated Structures. We demonstrate automated assembly using a dual-robot-arm system in Figure 6.8. The robots use the manipulation policy [120] and the asynchronous multi-agent planner [68] to manipulate toy bricks and construct the structure. Since the generated structures are physically stable, efficient and automated assembly can be performed.

Brick Texture and Color Generation. Figure 6.7 shows both UV texturing and uniform coloring results of brick structures, demonstrating our method's ability to generate diverse styles while preserving the underlying geometry.

6.6 Discussion

In this work, we have introduced BRICKGPT, an autoregressive model for generating interconnecting toy brick structures from text prompts. Our method learns to predict the next brick sequentially while ensuring physical stability and buildability. We have shown that our method outperforms LLM backbone models and several recent text-to-3D generation methods.

Limitations. Though our method outperforms existing methods, it still has several limitations. First, due to limited computational resources, we have not explored the largest 3D dataset. As a result, our method is restricted to producing designs within a $20 \times 20 \times 20$ grid across 21 categories, while recent 3D generation methods can create a wider variety of objects. Future work includes scaling up model training at higher grid resolutions on larger, more diverse datasets, such as Objaverse-XL [36]. Training on large-scale datasets can also improve generalization to out-of-distribution text prompts.

Second, our method currently supports a fixed set of commonly used toy bricks. In future work, we plan to expand the brick library to include a broader range of

dimensions and brick types, such as slopes and tiles, allowing for more diverse and intricate designs.

Chapter 7

Conclusions

7.1 Discussion

In this dissertation, we have explored the central challenge of democratizing 3D content creation. By leveraging powerful, data-driven priors, we can significantly lower the barriers that have traditionally confined 3D creation to the realm of experts. By developing models based on vast datasets of images, text, and shapes, we can empower everyday users to capture, generate, and realize 3D content from simple and intuitive inputs. The work presented here spans three parts, collectively addressing the fundamental aspects of: *geometry*, *appearance*, and *physics*.

Sparse-View 3D Reconstruction. We began by addressing the capture of real-world scenes. In Chapter 2, DS-NeRF [39] demonstrated that by using "free" depth priors from Structure-from-Motion, we can achieve high-fidelity 3D reconstruction from as few as two images, removing the need for dense, specialized data capture.

3D Asset Generation. We then turned to the creation of novel digital assets. In Chapters 3, 4, and 5, we introduced methods for generating complex 3D content from simple inputs. This included generating editable 3D objects from 2D sketches [40], using OctreeGPT [42] for efficient text-to-shape generation, and developing FlashTex [41] with LightControlNet to produce high-quality, relightable textures. These contributions make the creative process more accessible, replacing

complex manual modeling with intuitive commands.

Physical Asset Generation. Finally, we explored a novel direction by bridging the gap between digital design and physical reality. In Chapter 6, BrickGPT [181] showed that by integrating physics and manufacturing constraints, we can generate designs from text that are not only visually compelling but also structurally stable and buildable in the real world.

Collectively, we made small steps towards a future where 3D creation is as easy as writing a sentence or drawing a sketch. Meanwhile, this work also opens up new questions and reveals avenues for future exploration.

7.2 Future Work

While the methods in this thesis demonstrate considerable progress, several limitations point toward exciting areas for future research.

Unified Generative Models. This thesis largely treats the generation of shape, texture, and material as separate stages. While this modularity is practical, a key next step is to develop unified generative models that can create complete, textured, and relightable 3D assets in a single, end-to-end process. Such models could learn the complex interplay between geometry and appearance, leading to even more realistic and coherent results.

Large-scale 4D Scene Generation with Functional Assets. While Part II of this thesis has primarily focused on generating individual objects, a crucial next step is to scale these techniques to create large, dynamic 3D scenes. Our recent work, Cube [191], represents a preliminary step in this direction by using a large language model [2] as an agent to compose scenes from generated assets by techniques presented in Part II. However, the ultimate goal of generating expansive, functional worlds, e.g., a cyberpunk city populated with drivable vehicles and autonomous pedestrians, remains a significant and largely unexplored challenge. This will require not only scaling asset generation but also modeling the complex functionality, behaviors, layout and interaction of 3D assets that bring a virtual world to life.

Scaling and Generalization. The models presented were trained on large but finite datasets like Objaverse [37] and ShapeNet [19]. As a result, their ability to generalize to out-of-distribution objects or highly complex scenes is still limited. Scaling these methods to train on even larger and more diverse datasets, such as Objaverse-XL [37], is a crucial step for improving robustness. Furthermore, large video generative models, such as Google Veo 3, have demonstrated stunning results with reasonable 3D understanding. Exploring how to utilize 3D priors from video generative models could be helpful to expand the variety of 3D content that can be generated.

Advanced User Interaction and Control. While we have simplified the input to text and sketches, a significant area for future work lies in creating more user-friendly interactive systems. Imagine an interface where a user can iteratively refine a generated asset with a combination of text prompts ("make the legs thinner"), direct manipulation (pulling a vertex), and sketching (drawing a new handle). Developing real-time models that can seamlessly interpret these multimodal inputs would provide users with far more granular and intuitive control.

Generalizing Physics-Aware Generation. BrickGPT [181] represents a first step toward physically plausible generation, but it is constrained to a single domain of interlocking bricks. A grand challenge is to extend these ideas to a broader range of materials and fabrication processes. Future systems could generate designs optimized for 3D printing, considering material strength and printability constraints, or even design functional robotic parts or architectural plans with inherent structural integrity.

Bibliography

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. In *European Conference on Computer Vision (ECCV) Workshop*, 2022.

[4] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. Törf: Time-of-flight radiance fields for dynamic scene view synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[5] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[6] 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 *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[7] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. In *ACM SIGGRAPH*, 2019.

[8] David Benson and Joel Davis. Octree textures. In *ACM Transactions on Graphics (TOG)*, 2002.

[9] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Milos Hasan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint*

arXiv:2008.03824, 2020.

[10] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *International Conference on Learning Representations (ICLR)*, 2018.

[11] Alexey Bokhovkin, Shubham Tulsiani, and Angela Dai. Mesh2tex: Generating mesh textures from image queries. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[12] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

[13] Shengqu Cai, Anton Obukhov, Dengxin Dai, and Luc Van Gool. Pix2nerf: Unsupervised conditional π-gan for single image to neural radiance fields translation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[14] John Canny. A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1986.

[15] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[16] Caroline Chan, Frédo Durand, and Phillip Isola. Learning to generate line drawings that convey geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[17] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[18] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[19] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[20] Anpei Chen, Ruiyang Liu, Ling Xie, Zhang Chen, Hao Su, and Jingyi Yu. Sofgan: A portrait image generator with dynamic styling. In *ACM SIGGRAPH*,

2021.

[21] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[22] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022.

[23] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[24] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[25] Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[26] Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Yanru Wang, Zhibin Wang, Chi Zhang, Jingyi Yu, Gang Yu, Bin Fu, and Tao Chen. MeshXL: Neural coordinate field for generative 3D foundation models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[27] Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 3-sweep: Extracting editable objects from a single photo. In *ACM Transactions on Graphics (TOG)*, 2013.

[28] Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, et al. Meshanything: Artist-created mesh generation with autoregressive transformers. In *International Conference on Learning Representations (ICLR)*, 2024.

[29] Yiwen Chen, Yikai Wang, Yihao Luo, Zhengyi Wang, Zilong Chen, Jun Zhu, Chi Zhang, and Guosheng Lin. Meshanything v2: Artist-created mesh generation with adjacent mesh tokenization. In *IEEE International Conference on Computer Vision (ICCV)*, 2025.

[30] Yongwei Chen, Rui Chen, Jiabao Lei, Yabin Zhang, and Kui Jia. Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[31] Yuedong Chen, Qianyi Wu, Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai.

Sem2nerf: Converting single-view semantic masks to neural radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022.

[32] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis from sparse views of novel scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[33] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016.

[34] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[35] Morteza Daneshmand, Ahmed Helmi, Egils Avots, Fatemeh Noroozi, Fatih Alisinanoglu, Hasan Sait Arslan, Jelena Gorbova, Rain Eric Haamer, Cagri Ozcinar, and Gholamreza Anbarjafari. 3d scanning: A comprehensive survey. *arXiv preprint arXiv:1801.08863*, 2018.

[36] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-XL: A universe of 10M+ 3D objects. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[37] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[38] Johanna Delanoy, Adrien Bousseau, Mathieu Aubry, Phillip Isola, and Alexei A Efros. What you sketch is what you get: 3d sketching using multi-view deep volumetric prediction. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, 2018.

[39] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[40] Kangle Deng, Gengshan Yang, Deva Ramanan, and Jun-Yan Zhu. 3d-aware conditional image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[41] Kangle Deng, Timothy Omernick, Alexander Weiss, Deva Ramanan, Jun-Yan Zhu, Tinghui Zhou, and Maneesh Agrawala. Flashtex: Fast relightable mesh

texturing with lightcontrolnet. In *European Conference on Computer Vision (ECCV)*, 2024.

[42] Kangle Deng, Hsueh-Ti Derek Liu, Yiheng Zhu, Xiaoxia Sun, Chong Shang, Kiran Bhat, Deva Ramanan, Jun-Yan Zhu, Maneesh Agrawala, and Tinghui Zhou. Efficient autoregressive shape generation via octree-based adaptive tokenization. In *IEEE International Conference on Computer Vision (ICCV)*, 2025.

[43] Yuan Dong, Qi Zuo, Xiaodong Gu, Weihao Yuan, Zhengyi Zhao, Zilong Dong, Liefeng Bo, and Qixing Huang. GPLD3D: Latent diffusion of 3D shape generative models by enforcing geometric and physical priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[44] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[45] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[46] Pablo Funes and Jordan Pollack. Evolutionary body building: Adaptive physical designs for robots. *Artificial Life*, 1998.

[47] Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. In *ACM SIGGRAPH*, 2019.

[48] Ruihan Gao, Kangle Deng, Gengshan Yang, Wenzhen Yuan, and Jun-Yan Zhu. Tactile dreamfusion: Exploiting tactile sensing for 3d generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[49] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH*, 1997.

[50] Michael Garland and Yuan Zhou. Quadric-based simplification in any dimension. In *ACM Transactions on Graphics (TOG)*, 2005.

[51] Jiahao Ge, Mingjun Zhou, Wenrui Bao, Hao Xu, and Chi-Wing Fu. Creating LEGO figurines from single images. In *ACM Transactions on Graphics (TOG)*, 2024.

[52] Jiahao Ge, Mingjun Zhou, and Chi-Wing Fu. Learn to create simple LEGO micro buildings. In *ACM Transactions on Graphics (TOG)*, 2024.

[53] Andrew Goldberg, Kavish Kondap, Tianshuang Qiu, Zehan Ma, Letian Fu, Justin Kerr, Huang Huang, Kaiyuan Chen, Kuan Fang, and Ken Goldberg. Blox-Net: Generative design-for-robot-assembly using VLM supervision,

physics, simulation, and a robot with reset. In *IEEE International Conference on Robotics & Automation (ICRA)*, 2025.

[54] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[55] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations (ICLR)*, 2022.

[56] Minghao Guo, Bohan Wang, Pingchuan Ma, Tianyuan Zhang, Crystal Elaine Owens, Chuang Gan, Joshua B. Tenenbaum, Kaiming He, and Wojciech Matusik. Physically compatible 3D object modeling from a single image. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[57] Gurobi Optimization, LLC. Gurobi Optimizer reference manual, 2023. URL https://www.gurobi.com.

[58] Zekun Hao, David W Romero, Tsung-Yi Lin, and Ming-Yu Liu. Meshtron: High-fidelity, artist-like 3d mesh generation at scale. *arXiv preprint arXiv:2412.09548*, 2024.

[59] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[60] Paul Henderson, Vagia Tsiminaki, and Christoph Lampert. Leveraging 2D data to learn textured 3D mesh generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[61] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[62] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

[63] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[64] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[65] Fangzhou Hong, Jiaxiang Tang, Ziang Cao, Min Shi, Tong Wu, Zhaoxi Chen,

Tengfei Wang, Liang Pan, Dahua Lin, and Ziwei Liu. 3DTopia: Large text-to-3D generation model with hybrid diffusion priors. *arXiv preprint arXiv:2403.02234*, 2024.

[66] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. In *International Conference on Learning Representations (ICLR)*, 2024.

[67] Hsin-Ping Huang, Hung-Yu Tseng, Hsin-Ying Lee, and Jia-Bin Huang. Semantic view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[68] Philip Huang, Ruixuan Liu, Shobhit Aggarwal, Changliu Liu, and Jiaoyang Li. Apex-mr: Multi-robot asynchronous planning and execution for cooperative assembly. In *Robotics: Science and Systems*, 2025.

[69] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3D scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[70] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *European Conference on Computer Vision (ECCV)*, 2018.

[71] Yukun Huang, Jianan Wang, Yukai Shi, Xianbiao Qi, Zheng-Jun Zha, and Lei Zhang. Dreamtime: An improved optimization strategy for text-to-3d content creation. In *International Conference on Learning Representations (ICLR)*, 2025.

[72] Zeng Huang, Tianye Li, Weikai Chen, Yajie Zhao, Jun Xing, Chloe Legendre, Linjie Luo, Chongyang Ma, and Hao Li. Deep volumetric video from very sparse multi-view performance capture. In *European Conference on Computer Vision (ECCV)*, 2018.

[73] Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *International Conference on Machine Learning (ICML)*, 2023.

[74] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH*, 1999.

[75] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[76] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST)*, 2011.

[77] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning (ICML)*, 2021.

[78] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[79] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[80] Kaiwen Jiang, Shu-Yu Chen, Feng-Lin Liu, Hongbo Fu, and Lin Gao. Nerffaceediting: Disentangled face editing in neural radiance fields. In *ACM SIGGRAPH Asia*, 2022.

[81] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, 2016.

[82] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[83] Tao Ju, Frank Losasso, Scott Schaefer, and Joe D. Warren. Dual contouring of hermite data. In *ACM Transactions on Graphics (TOG)*, 2002.

[84] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. In *ACM SIGGRAPH*, 1984.

[85] Takeo Kanade, PJ Narayanan, and Peter W Rander. Virtualized reality: Concepts and early results. In *Proceedings IEEE Workshop on Representation of Visual Scenes*, 1995.

[86] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[87] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[88] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[89] Oren Katzir, Or Patashnik, Daniel Cohen-Or, and Dani Lischinski. Noise-free score distillation. In *International Conference on Learning Representations (ICLR)*, 2024.

[90] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. In *ACM Transactions on Graphics (TOG)*, 2023.

[91] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3d object manipulation in a single photograph using stock 3d models. In *ACM Transactions on Graphics (TOG)*, 2014.

[92] Jae Woo Kim. Survey on automated LEGO assembly construction. In *International Conference on Advanced Communication Technology (ICACT)*, 2014.

[93] Jungtaek Kim, Hyunsoo Chung, Jinhwi Lee, Minsu Cho, and Jaesik Park. Combinatorial 3D shape generation via sequential assembly. *arXiv preprint arXiv:2004.07414*, 2020.

[94] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[95] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[96] Black Forest Labs. Flux. https://huggingface.co/black-forest-labs/FLUX.1-schnell, 2024.

[97] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. In *ACM SIGGRAPH*, 2020.

[98] LDraw.org. Ldraw.org homepage, 2025. URL https://www.ldraw.org/.

[99] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[100] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[101] Kyle Lennon, Katharina Fransen, Alexander O'Brien, Yumeng Cao, Matthew Beveridge, Yamin Arefeen, Nikhil Singh, and Iddo Drori. Image2Lego: Customized LEGO set generation from images. *arXiv preprint arXiv:2108.08477*,

2021.

[102] Marc Levoy and Pat Hanrahan. Light field rendering. In *ACM SIGGRAPH*, 1996.

[103] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. In *International Conference on Learning Representations (ICLR)*, 2024.

[104] Muheng Li, Yueqi Duan, Jie Zhou, and Jiwen Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[105] Weiyu Li, Rui Chen, Xuelin Chen, and Ping Tan. Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. In *International Conference on Learning Representations (ICLR)*, 2024.

[106] Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. Craftsman3d: High-fidelity mesh generation with 3d native generation and interactive geometry refiner. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[107] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[108] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[109] Zhengqin Li, Kalyan Sunkavalli, and Manmohan Chandraker. Materials for masses: Svbrdf acquisition with a single mobile phone image. In *European Conference on Computer Vision (ECCV)*, 2018.

[110] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[111] 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. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[112] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Com-

mon objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[113] Yehonathan Litman, Or Patashnik, Kangle Deng, Aviral Agrawal, Rushikesh Zawar, Fernando De la Torre, and Shubham Tulsiani. Materialfusion: Enhancing inverse rendering with material diffusion priors. In *International Conference on 3D Vision (3DV)*, 2025.

[114] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[115] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[116] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[117] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[118] Minghua Liu, Chong Zeng, Xinyue Wei, Ruoxi Shi, Linghao Chen, Chao Xu, Mengqi Zhang, Zhaoning Wang, Xiaoshuai Zhang, Isabella Liu, et al. Meshformer: High-quality mesh generation with 3d-guided reconstruction model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[119] Ruixuan Liu, Kangle Deng, Ziwei Wang, and Changliu Liu. Stablelego: Stability analysis of block stacking assembly. In *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[120] Ruixuan Liu, Yifan Sun, and Changliu Liu. A lightweight and transferable design for robust LEGO manipulation. *International Symposium on Flexible Automation*, 2024.

[121] Ruixuan Liu, Alan Chen, Weiye Zhao, and Changliu Liu. Physics-aware combinatorial assembly sequence planning using data-free action masking. In *IEEE Robotics and Automation Letters (RA-L)*, 2025.

[122] 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 *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[123] 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 *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[124] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[125] Xueyi Liu, Bin Wang, He Wang, and Li Yi. Few-shot physically-aware articulated mesh generation via hierarchical deformation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[126] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[127] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[128] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[129] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, 1998.

[130] Artem Lukoianov, Haitz Sáez de Ocáriz Borde, Kristjan Greenewald, Vitor Campagnolo Guizilini, Timur Bagautdinov, Vincent Sitzmann, and Justin Solomon. Score distillation via reparametrized ddim. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[131] Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 3d shape reconstruction from sketches via multi-view convolutional networks. In *International Conference on 3D Vision (3DV)*, 2017.

[132] Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. Legolization: optimizing LEGO designs. In *ACM Transactions on Graphics (TOG)*, 2015.

[133] Liane Makatura, Michael Foshey, Bohan Wang, Felix HähnLein, Pingchuan Ma, Bolei Deng, Megan Tjandrasuwita, Andrew Spielberg, Crystal Elaine Owens, Peter Yichen Chen, et al. How can large language models help humans in design and manufacturing? *arXiv preprint arXiv:2307.14377*, 2023.

[134] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[135] David McAllister, Songwei Ge, Jia-Bin Huang, David W. Jacobs, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Rethinking score distillation as a bridge between image distributions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[136] Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 1982.

[137] Donald JR Meagher. *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. Electrical and Systems Engineering Department Rensseiaer Polytechnic, 1980.

[138] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2022.

[139] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. Gnerf: Gan-based neural radiance field without posed camera. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[140] Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. Lt3sd: Latent trees for 3d scene diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[141] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, 2018.

[142] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[143] Moustafa Meshry, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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

[145] Mariem Mezghanni, Malika Boulkenafed, Andre Lieutier, and Maks Ovsjanikov. Physically-aware generative network for 3D shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[146] Mariem Mezghanni, Théo Bodrito, Malika Boulkenafed, and Maks Ovsjanikov. Physical simulation layer for accurate 3D modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

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

[148] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[149] 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 *European Conference on Computer Vision (ECCV)*, 2020.

[150] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[151] Vihaan Misra, Peter Schaldenbrand, and Jean Oh. ShapeShift: Towards text-to-shape arrangement synthesis with content-aware geometric constraints. *arXiv preprint arXiv:2503.14720*, 2025.

[152] Chong Mou, Xintao Wang, Liangbin Xie, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023.

[153] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *ACM SIGGRAPH*, 2022.

[154] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[155] Stylianos Mystakidis. Metaverse. *Encyclopedia*, 2022.

[156] Gimin Nam, Mariem Khlifi, Andrew Rodriguez, Alberto Tono, Linqi Zhou, and Paul Guerrero. 3d-ldm: Neural implicit 3d shape generation with latent diffusion models. *arXiv preprint arXiv:2212.00842*, 2022.

[157] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter Battaglia. PolyGen: An autoregressive generative model of 3D meshes. In *International Conference on Machine Learning (ICML)*, 2020.

[158] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum (EGSR)*, 2021.

[159] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[160] Junfeng Ni, Yixin Chen, Bohan Jing, Nan Jiang, Bin Wang, Bo Dai, Puhao Li, Yixin Zhu, Song-Chun Zhu, and Siyuan Huang. PhyRecon: Physically plausible neural scene reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[161] Fred E Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied optics*, 1965.

[162] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[163] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[164] Sumiaki Ono, Alexis Andre, Youngha Chang, and Masayuki Nakajima. LEGO builder: Automatic generation of LEGO assembly manual from 3D polygon model. *ITE Transactions on Media Technology and Applications*, 2013.

[165] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[166] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. Stylesdf: High-resolution 3d-consistent image

and geometry generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[167] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th annual symposium on user interface software and technology (UIST)*, 2016.

[168] Xingang Pan, Xudong Xu, Chen Change Loy, Christian Theobalt, and Bo Dai. A shading-guided generative implicit model for shape-accurate 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[169] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[170] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[171] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. In *ACM SIGGRAPH Asia*, 2018.

[172] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[173] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[174] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[175] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision (ECCV)*, 2020.

[176] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[177] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[178] Dario Pavllo, Jonas Kohler, Thomas Hofmann, and Aurelien Lucchi. Learning generative models of textured 3d meshes from real-world images. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[179] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.

[180] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[181] Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. Generating physically stable and buildable brick structures from text. In *IEEE International Conference on Computer Vision (ICCV)*, 2025.

[182] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[183] 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 *International Conference on Machine Learning (ICML)*, 2021.

[184] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. Dreambooth3d: Subject-driven text-to-3d generation. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[185] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[186] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[187] Fabio Remondino. Heritage recording and 3d modeling with photogrammetry and 3d scanning. *Remote sensing*, 2011.

[188] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *IEEE Conference on Computer Vision and Pattern Recogni-*

tion (CVPR), 2024.

[189] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *ACM SIGGRAPH*, 2023.

[190] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[191] Foundation AI Team Roblox. Cube: A roblox view of 3d intelligence. *arXiv preprint arXiv:2503.15475*, 2025.

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

[193] Nataniel Ruiz, Eunji Chong, and James M. Rehg. Fine-grained head pose estimation without keypoints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2018.

[194] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[195] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[196] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[197] Sam Sartor and Pieter Peers. Matfusion: a generative diffusion model for svbrdf capture. In *ACM SIGGRAPH Asia*, 2023.

[198] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. In *ACM SIGGRAPH*, 2006.

[199] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[200] Edgar Schönfeld, Vadim Sushko, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. In *International Conference on Learning Representations (ICLR)*,

2020.

[201] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[202] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[203] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[204] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *International Conference on Learning Representations (ICLR)*, 2024.

[205] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[206] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[207] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces. In *European Conference on Computer Vision (ECCV)*, 2022.

[208] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. MeshGPT: Generating triangle meshes with decoder-only transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19615–19625, 2024.

[209] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[210] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[211] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathain, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

[212] Chonghyuk Song, Gengshan Yang, Kangle Deng, Jun-Yan Zhu, and Deva Ramanan. Total-recon: Deformable scene reconstruction for embodied view synthesis. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[213] Zhuo Su, Wenzhe Liu, Zitong Yu, Dewen Hu, Qing Liao, Qi Tian, Matti Pietikainen, and Li Liu. Pixel difference networks for efficient edge detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[214] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[215] Jingxiang Sun, Xuan Wang, Yichun Shi, Lizhen Wang, Jue Wang, and Yebin Liu. Ide-3d: Interactive disentangled editing for high-resolution 3d-aware portrait synthesis. In *ACM Transactions on Graphics (TOG)*, 2022.

[216] Jingxiang Sun, Xuan Wang, Yong Zhang, Xiaoyu Li, Qi Zhang, Yebin Liu, and Jue Wang. Fenerf: Face editing in neural radiance fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[217] Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. Dreamcraft3d: Hierarchical 3d generation with bootstrapped diffusion prior. In *International Conference on Learning Representations (ICLR)*, 2024.

[218] Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image understanding*, 1993.

[219] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: A survey from 2010 to 2016. *IPSJ transactions on computer vision and applications*, 2017.

[220] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[221] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[222] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In

International Conference on Learning Representations (ICLR), 2024.

[223] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. LGM: Large multi-view gaussian model for high-resolution 3D content creation. In *European Conference on Computer Vision (ECCV)*, 2025.

[224] Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. In *International Conference on Learning Representations (ICLR)*, 2025.

[225] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[226] Romain Pierre Testuz, Yuliy Schwartzburg, and Mark Pauly. Automatic generation of constructable brick sculptures. Technical report, École Polytechnique Fédérale de Lausanne, 2013.

[227] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[228] Rylee Thompson, Elahe Ghalebi, Terrance DeVries, and Graham W Taylor. Building lego using deep generative models of graphs. *arXiv preprint arXiv:2012.11543*, 2020.

[229] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[230] TobyLobster. ImportLDraw - blender add-on for importing LDraw models. https://github.com/TobyLobster/ImportLDraw, 2025.

[231] Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, , Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. Triposr: Fast 3d object reconstruction from a single image. *arXiv preprint arXiv:2403.02151*, 2024.

[232] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. In *ACM Transactions on Graphics (TOG)*, 2021.

[233] Uy Dieu Tran, Minh Luu, Phong Ha Nguyen, Khoi Nguyen, and Binh-Son Hua. Diverse text-to-3d synthesis with augmented text embedding. In *European Conference on Computer Vision (ECCV)*, 2024.

[234] Richard Tucker and Noah Snavely. Single-view view synthesis with multi-

plane images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[235] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[236] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[237] Giuseppe Vecchio, Renato Sortino, Simone Palazzo, and Concetto Spampinato. Matfuse: Controllable material generation with diffusion models. *arXiv preprint arXiv:2308.11408*, 2023.

[238] Giuseppe Vecchio, Rosalie Martin, Arthur Roullier, Adrien Kaiser, Romain Rouffet, Valentin Deschaintre, and Tamy Boubekeur. Controlmat: Controlled generative approach to material capture. In *ACM Transactions on Graphics (TOG)*, 2024.

[239] Alexander Vilesov, Pradyumna Chari, and Achuta Kadambi. CG3D: Compositional generation for text-to-3D via Gaussian splatting. *arXiv preprint arXiv:2311.17907*, 2023.

[240] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

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

[242] Peng-Shuai Wang. Octformer: Octree-based transformers for 3D point clouds. In *ACM Transactions on Graphics (TOG)*, 2023.

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

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

[245] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[246] Ruocheng Wang, Yunzhi Zhang, Jiayuan Mao, Chin-Yi Cheng, and Jiajun Wu. Translating a visual lego manual to a machine-executable plan. In *European Conference on Computer Vision (ECCV)*, 2022.

[247] Tengfei Wang, Ting Zhang, Bo Zhang, Hao Ouyang, Dong Chen, Qifeng Chen, and Fang Wen. Pretraining is all you need for image-to-image translation. *arXiv preprint arXiv:2205.12952*, 2022.

[248] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[249] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[250] Zhengyi Wang, Jonathan Lorraine, Yikai Wang, Hang Su, Jun Zhu, Sanja Fidler, and Xiaohui Zeng. Llama-mesh: Unifying 3d mesh generation with language models. *arXiv preprint arXiv:2411.09595*, 2024.

[251] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. In *IEEE Transactions on Image Processing (TIP)*, 2004.

[252] Zian Wang, Jonah Philion, Sanja Fidler, and Jan Kautz. Learning indoor inverse rendering with 3d spatially-varying lighting. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[253] Ethan Weber, Aleksander Holynski, Varun Jampani, Saurabh Saxena, Noah Snavely, Abhishek Kar, and Angjoo Kanazawa. Nerfiller: Completing scenes via generative 3d inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[254] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[255] Haohan Weng, Zibo Zhao, Biwen Lei, Xianghui Yang, Jian Liu, Zeqiang Lai, Zhuo Chen, Yuhong Liu, Jie Jiang, Chunchao Guo, et al. Scaling mesh generation via compressive tokenization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[256] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[257] Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3d: Scalable image-to-3d generation via 3d latent diffusion transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[258] Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, et al. Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation. In *ACM Transactions on Graphics (TOG)*, 2024.

[259] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[260] Xiaohua Xie, Kai Xu, Niloy J Mitra, Daniel Cohen-Or, Wenyong Gong, Qi Su, and Baoquan Chen. Sketch-to-design: Context-based part assembly. In *Computer Graphics Forum*, 2013.

[261] Bojun Xiong, Si-Tong Wei, Xin-Yang Zheng, Yan-Pei Cao, Zhouhui Lian, and Peng-Shuai Wang. Octfusion: Octree-based diffusion models for 3d shape generation. *arXiv preprint arXiv:2408.14732*, 2024.

[262] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[263] Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, et al. Vr-nerf: High-fidelity virtualized walkable spaces. In *ACM SIGGRAPH Asia*, 2023.

[264] Qingshan Xu, Jiao Liu, Melvin Wong, Caishun Chen, and Yew-Soon Ong. Precise-physics driven text-to-3D generation. *arXiv preprint arXiv:2403.12438*, 2024.

[265] Xudong Xu, Zhaoyang Lyu, Xingang Pan, and Bo Dai. Matlaber: Material-aware text-to-3d via latent brdf auto-encoder. *arXiv preprint arXiv:2308.09278*, 2023.

[266] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3d-aware image synthesis via learning structural and textural representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[267] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. In

International Conference on Learning Representations (ICLR), 2024.

[268] Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. PhyScene: Physically interactable 3D scene synthesis for embodied AI. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[269] Shunyu Yao, Tzu Ming Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, Bill Freeman, and Josh Tenenbaum. 3d-aware scene manipulation via inverse graphics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[270] Junliang Ye, Fangfu Liu, Qixiu Li, Zhengyi Wang, Yikai Wang, Xinzhou Wang, Yueqi Duan, and Jun Zhu. Dreamreward: Text-to-3d generation with human preference. In *European Conference on Computer Vision (ECCV)*, 2024.

[271] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[272] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[273] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. Pixelnerf: Neural radiance fields from one or few images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[274] Rui Yu, Yue Dong, Pieter Peers, and Xin Tong. Learning texture generators for 3d shape collections from internet photo sets. In *The British Machine Vision Conference (BMVC)*, 2021.

[275] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. PhysDiff: Physics-guided human motion diffusion model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[276] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[277] Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, BIN FU, Yong Liu, and Gang Yu. Paint3d: Paint anything 3d with lighting-less texture diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[278] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. In *ACM Transactions on Graphics (TOG)*, 2023.

[279] Jichao Zhang, Enver Sangineto, Hao Tang, Aliaksandr Siarohin, Zhun Zhong, Nicu Sebe, and Wei Wang. 3d-aware semantic-guided generative model for human synthesis. In *European Conference on Computer Vision (ECCV)*, 2022.

[280] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[281] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024.

[282] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. In *ACM Transactions on Graphics (TOG)*, 2024.

[283] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[284] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[285] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[286] Minda Zhao, Chaoyi Zhao, Xinyue Liang, Lincheng Li, Zeng Zhao, Zhipeng Hu, Changjie Fan, and Xin Yu. Efficientdreamer: High-fidelity and robust 3d creation via orthogonal-view diffusion prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[287] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[288] Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

[289] Guyue Zhou, Liyi Luo, Hao Xu, Xinliang Zhang, Haole Guo, and Hao Zhao. Brick yourself within 3 minutes. In *IEEE International Conference on Robotics & Automation (ICRA)*, 2022.

[290] J. Zhou, X. Chen, and Y. Xu. Automatic generation of vivid LEGO architectural sculptures. *Computer Graphics Forum*, 2019.

[291] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[292] Mingjun Zhou, Jiahao Ge, Hao Xu, and Chi-Wing Fu. Computational design of LEGO® sketch art. In *ACM Transactions on Graphics (TOG)*, 2023.

[293] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European Conference on Computer Vision (ECCV)*, 2020.

[294] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision (ECCV)*, 2016.

[295] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[296] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[297] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. Visual object networks: Image generation with disentangled 3d representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[298] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[299] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[300] Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *ACM SIGGRAPH Asia*, 2023.

[301] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.