Advances in 3D Generation: A Survey

Xiaoyu Li\textsuperscript{1} Qi Zhang\textsuperscript{1} Di Kang\textsuperscript{1} Weihao Cheng\textsuperscript{2} Yiming Gao\textsuperscript{2} Jingbo Zhang\textsuperscript{3} Zhihao Liang\textsuperscript{4} Jing Liao\textsuperscript{3} Yan-Pei Cao\textsuperscript{1,2} Ying Shan\textsuperscript{1,2\dagger}\\
\textsuperscript{1}Tencent AI Lab  \textsuperscript{2}ARC Lab, Tencent PCG  \textsuperscript{3}City University of Hong Kong  \textsuperscript{4}South China University of Technology  
\textsuperscript{⋆}Equal contribution. \textsuperscript{†}Corresponding author.

Figure 1: In this survey, we investigate a large variety of 3D generation methods. Over the past decade, 3D generation has achieved remarkable progress and has recently garnered considerable attention due to the success of generative AI in images and videos. 3D generation results from 3D-GAN [WZX\textsuperscript{*}16], DeepSDF [PFS\textsuperscript{*}19], DMTet [SGY\textsuperscript{*}21], EG3D [CLC\textsuperscript{*}22], DreamFusion [PJBM23], PointE [NJD\textsuperscript{*}22], Zero-1-to-3 [LWVH\textsuperscript{*}23] and Instant3D [LTZ\textsuperscript{*}23].

Abstract
Generating 3D models lies at the core of computer graphics and has been the focus of decades of research. With the emergence of advanced neural representations and generative models, the field of 3D content generation is developing rapidly, enabling the creation of increasingly high-quality and diverse 3D models. The rapid growth of this field makes it difficult to stay abreast of all recent developments. In this survey, we aim to introduce the fundamental methodologies of 3D generation methods and establish a structured roadmap, encompassing 3D representation, generation methods, datasets, and corresponding applications. Specifically, we introduce the 3D representations that serve as the backbone for 3D generation. Furthermore, we provide a comprehensive overview of the rapidly growing literature on generation methods, categorized by the type of algorithmic paradigms, including feedforward generation, optimization-based generation, procedural generation, and generative novel view synthesis. Lastly, we discuss available datasets, applications, and open challenges. We hope this survey will help readers explore this exciting topic and foster further advancements in the field of 3D content generation.

1. Introduction
Automatically generating 3D models using algorithms has long been a significant task in computer vision and graphics. This task has garnered considerable interest due to its broad applications in video games, movies, virtual characters, and immersive experiences, which typically require a wealth of 3D assets. Recently, the success of neural representations, particularly Neural Radiance Fields (NeRFs) [MST\textsuperscript{*}20, BMT\textsuperscript{*}21, MESK22, KKL23], and generative models such as diffusion models [HJA20, RBL\textsuperscript{*}22a], has led to remarkable advancements in 3D content generation.
In the realm of 2D content generation, recent advancements in generative models have steadily enhanced the capacity for image generation and editing, leading to increasingly diverse and high-quality results. Pioneering research on generative adversarial networks (GANs) [GPAM+14, AQW19], variational autoencoders (VAEs) [KPHL17, PGH†16, KW13], and autoregressive models [RWC†19, BMR‡20] has demonstrated impressive outcomes. Furthermore, the advent of generative artificial intelligence (AI) and diffusion models [HJA20, ND21, SCS†22] signifies a paradigm shift in image manipulation techniques, such as Stable Diffusion [RBL†22a], Imagen [SCS†22], Midjourney [Mid], or DALL-E 3 [Ope]. These generative AI models enable the creation and editing of photorealistic or stylized images, or even videos [CZC†24, HSG†22, SPH†23, GNL†23], using minimal input like text prompts. As a result, they often generate imaginative content that transcends the boundaries of the real world, pushing the limits of creativity and artistic expression. Owing to their “emergent” capabilities, these models have redefined the limits of what is achievable in content generation, expanding the horizons of creativity and artistic expression.

The demand to extend 2D content generation into 3D space is becoming increasingly crucial for applications in generating 3D assets or creating immersive experiences, particularly with the rapid development of the metaverse. The transition from 2D to 3D content generation, however, is not merely a technological evolution. It is primarily a response to the demands of modern applications that necessitate a more intricate replication of the physical world, which 2D representations often fail to provide. This shift highlights the limitations of 2D content in applications that require a comprehensive understanding of spatial relationships and depth perception.

As the significance of 3D content becomes increasingly evident, there has been a surge in research efforts dedicated to this domain. However, the transition from 2D to 3D content generation is not a straightforward extension of existing 2D methodologies. Instead, it involves tackling unique challenges and re-evaluating data representation, formulation, and underlying generative models to effectively address the complexities of 3D space. For instance, it is not obvious how to integrate the 3D scene representations into 2D generative models to handle higher dimensions, as required for 3D generation. Unlike images or videos which can be easily collected from the web, 3D assets are relatively scarce. Furthermore, evaluating the quality of generated 3D models presents additional challenges, as it is necessary to develop better formulations for objective functions, particularly when considering multi-view consistency in 3D space. These complexities demand innovative approaches and novel solutions to bridge the gap between 2D and 3D content generation.

While not as prominently featured as its 2D counterpart, 3D content generation has been steadily progressing with a series of notable achievements. The representative examples shown in Fig. 1 demonstrate significant improvements in both quality and diversity, transitioning from early methods like 3D-GAN [WZX†16] to recent approaches like Instant3D [LTZ†23]. Therefore, this survey paper seeks to systematically explore the rapid advancements and multifaceted developments in 3D content generation. We present a structured overview and comprehensive roadmap of the many recent works focusing on 3D representations, 3D generation methods, datasets, and applications of 3D content generation, and to outline open challenges.

Fig. 2 presents an overview of this survey. We first discuss the scope and related work of this survey in Sec. 2. In the following sections, we examine the core methodologies that form the foundation of 3D content generation. Sec. 3 introduces the primary scene representations and their corresponding rendering functions used in 3D content generation. Sec. 4 explores a wide variety of 3D generation methods, which can be divided into four categories based on their algorithmic methodologies: feedforward generation, optimization-based generation, procedural generation, and generative novel view synthesis. An evolutionary tree of these methodologies is also depicted to illustrate their primary branch. As data accumulation plays a vital role in ensuring the success of deep learning models, we present related datasets employed for training 3D generation methods. In the end, we include a brief discussion on related applications, such as 3D human and face generation, outline open challenges, and conclude this survey. We hope this survey offers a systematic summary of 3D generation that could inspire subsequent work for interested readers.

In this work, we present a comprehensive survey on 3D generation, with two main contributions:

- Given the recent surge in contributions based on generative models in the field of 3D vision, we provide a comprehensive and timely literature review of 3D content generation, aiming to offer readers a rapid understanding of the 3D generation framework and its underlying principles.
- We propose a multi-perspective categorization of 3D generation methods, aiming to assist researchers working on 3D content generation in specific domains to quickly identify relevant works and facilitate a better understanding of the related techniques.

2. Scope of This Survey

In this survey, we concentrate on the techniques for the generation of 3D models and their related datasets and applications. Specifically, we first give a short introduction to the scene representation. Our focus then shifts to the integration of these representations and the generative models. Then, we provide a comprehensive overview of the prominent methodologies of generation methods. We also explore the related datasets and cutting-edge applications such as 3D human generation, 3D face generation, and 3D editing, all of which are enhanced by these techniques.

This survey is dedicated to systematically summarizing and categorizing 3D generation methods, along with the related datasets and applications. The surveyed papers are mostly published in major computer vision and computer graphics conferences/journals as well as some preprints released on arXiv in 2023. While it’s challenging to exhaust all methods related to 3D generation, we hope to include as many major branches of 3D generation as possible. We do not delve into detailed explanations for each branch, instead, we typically introduce some representative works within it to explain its paradigm. The details of each branch can be found in the related work section of these cited papers.

Related Survey. Neural reconstruction and rendering with scene
representations are closely related to 3D generation. However, we consider these topics to be outside the purview of this report. For a comprehensive discussion on neural rendering, we direct readers to [TFT∗20, TTM∗22], and for a broader examination of other neural representations, we recommend [KBM∗20, XTS∗22]. Our primary focus is on exploring techniques that generate 3D models. Therefore, this review does not encompass research on generation methods for 2D images within the realm of visual computing. For further information on a specific generation method, readers can refer to [Doe16] (VAEs), [GSW∗21] (GANs), [PYG∗23, CHIS23] (Diffusion) and [KNH∗22] (Transformers) for a more detailed understanding. There are also some surveys related to 3D generation that have their own focuses such as 3D-aware image synthesis [XX23], 3D generative models [SPX∗22], Text-to-3D [LZW∗23] and deep learning for 3D point clouds [GWH∗20]. In this survey, we give a comprehensive analysis of different 3D generation methods.

3. Neural Scene Representations

In the domain of 3D AI-generated content, adopting a suitable representation of 3D models is essential. The generation process typically involves a scene representation and a differentiable rendering algorithm for creating 3D models and rendering 2D images. Conversely, the created 3D models or 2D images could be supervised in the reconstruction domain or image domain, as illustrated in Fig. 3. Some methods directly supervise the 3D models of the scene representation, while others render the scene representation into images and supervise the resulting renderings. In the following, we broadly classify the scene representations into three groups: explicit scene representations (Section 3.1), implicit representations (Section 3.2), and hybrid representations (Section 3.3). Note that, the rendering methods (e.g., ray casting, volume rendering, rasterization, etc.) which should be differentiable to optimize the scene representations from various inputs, are also introduced.

3.1. Explicit Representations

Explicit scene representations serve as a fundamental module in computer graphics and vision, as they offer a comprehensive means of describing 3D scenes. By depicting scenes as an assembly of basic primitives, including point-like primitives, triangle-based meshes, and advanced parametric surfaces, these representations can create detailed and accurate visualizations of various environments and objects.

3.1.1. Point Clouds

A point cloud is a collection of elements in Euclidean space, representing discrete points with addition attributes (e.g., colors and normals) in three-dimensional space. In addition to simple points, which can be considered infinitesimally small surface patches, oriented point clouds with a radius (surfels) can also be used [PZVBG00]. Surfels are used in computer graphics for rendering
Figure 3: Neural scene representations used for 3D generation, including explicit, implicit, and hybrid representations. The 3D generation involves the use of scene representations and a differentiable rendering algorithm to create 3D models or render 2D images. On the flip side, these 3D models or 2D images can function as the reconstruction domain or image domain, overseeing the 3D generation of scene representations.

By incorporating point cloud-based differentiable renderers into the 3D generation process, researchers can leverage the benefits of point clouds while maintaining compatibility with gradient-based optimization techniques. This process can be generally categorized into two different ways: point splitting which blends the discrete samples with some local deterministic blurring kernels [ZPVBG02, LKL18, ID18, RROG18], and conventional point renderer [ASK20, DZL20, KPLD21, RALB22]. These methods facilitate the generation and manipulation of 3D point cloud models while maintaining differentiability, which is essential for training and optimizing neural networks in 3D generation tasks.

3.1.2. Meshes

By connecting multiple vertices with edges, more complex geometric structures (e.g. wireframes and meshes) can be formed [BKP10]. These structures can then be further refined by using polygons, typically triangles or quadrilaterals, to create realistic representations of objects [SS87]. Meshes provide a versatile and efficient means of representing intricate shapes and structures, as they can be easily manipulated and rendered by computer algorithms. The majority of graphic editing toolchains utilize triangle meshes. This type of representation is indispensable for any digital content creation (DCC) pipeline, given its wide acceptance and compatibility. To align seamlessly with these pipelines, neural networks can be strategically trained to predict discrete vertex locations [BNT21, TZN19]. This ability allows for the direct importation of these locations into any DCC pipeline, facilitating a smooth...
and efficient workflow. In contrast to predicting discrete textures, continuous texture methods optimized via neural networks are proposed, such as texture fields [OMN*19] and NeRF-Tex [BGP*22]. In this way, it could provide a more refined and detailed texture, enhancing the overall quality and realism of the generated 2D models.

Integrating mesh representation into 3D generation requires the use of mesh-based differentiable rendering methods, which enable meshes to be rasterized in a manner that is compatible with gradient-based optimization. Several such techniques have been proposed, including OpenDR [LB14], neural mesh renderer [KUH18], Paparazzi [LTJ18], and Soft Rasterizer [LLCL19]. Additionally, general-purpose physically based renderers like Mitsuba [KUH18], Paparazzi [LTJ18], and Soft Rasterizer [LLCL19]. The importance of mesh representation in 3D generation is evident as it serves as a fundamental component in the rendering pipeline, enabling the seamless integration of differentiable rendering methods with traditional rendering techniques.

3.1.3. Multi-layer Representations

The use of multiple semi-transparent colored layers for representing scenes has been a popular and successful approach in real-time novel view synthesis [ZTF*18]. Using Layered Depth Image (LDI) representation [SGHS98] is a notable example, extending traditional depth maps by incorporating multiple layers of depth maps, each with associated color values. Several methods [PZ17, CBT*19, SSKH20] have drawn inspiration from the LDI representation and employed deep learning advancements to create networks capable of predicting LDI's. In addition to LDI's, Stereomagnification [ZTF*18] initially introduced the multiple image (MPI) representation. It describes scenes using multiple frontparallel semi-transparent layers, including colors and opacity, at fixed depths ranging through plane sweep volumes. With the help of volume rendering and homography projection, the novel view could be synthesized in real-time. Building on Stereomagnification [ZTF*18], various methods [FBD*19, MSOC*19, STB*19] have adopted the MPI representation to enhance rendering quality. The multi-layer representation has been further expanded to accommodate wider fields of view in [BFO*20, ALG*20, LXM*20] by substituting planes with spheres. As research in this domain continues to evolve, we can expect further advancements in these methods, leading to more efficient and effective 3D generation techniques for real-time rendering.

3.2. Implicit Representations

Implicit representations have become the scene representation of choice for problems in view synthesis or shape reconstruction, as well as many other applications across computer graphics and vision. Unlike explicit scene representations that usually focus on object surfaces, implicit representations could define the entire volume of a 3D object, and use volume rendering for image synthesis. These representations utilize mathematical functions, such as radiance fields [MST*20] or signed distance fields [PPS*19, CZ19], to describe the properties of a 3D space.

3.2.1. Neural Radiance Fields

Neural Radiance Fields (NeRFs) [MST*20] have gained prominence as a favored scene representation method for a wide range of applications. Fundamentally, NeRFs introduce a novel representation of 3D scenes or geometries. Rather than utilizing point clouds and meshes, NeRFs depict the scene as a continuous volume. This approach involves obtaining volumetric parameters, such as viewpoint and volume density, by querying an implicit neural network. This innovative representation offers a more fluid and adaptable way to capture the intricacies of 3D scenes, paving the way for enhanced rendering and modeling techniques.

Specifically, NeRF [MST*20] represents the scene with a continuous volumetric radiance field, which utilizes MLPs to map the position \( x \) and view direction \( r \) to a density \( \sigma \) and color \( c \). To render a pixel’s color, NeRF casts a single ray \( r(t) = x + t \mathbf{d} \) and evaluates a series of points \( \{t_i\} \) along the ray. The evaluated \( \{\langle \sigma_i, c_i \rangle\} \) at the sampled points are accumulated into the color \( C(r) \) of the pixel via volume rendering [Max95]:

\[
C(r) = \sum_i T_i c_i, \quad \text{where } T_i = \exp \left( -\sum_{k=0}^{i-1} \sigma_k \delta_k \right),
\]

and \( \alpha_i = 1 - \exp(-\sigma_i \delta_i) \) indicates the opacity of the sampled point. Accumulated transmittance \( T_i \) quantifies the probability of the ray traveling from \( t_0 \) to \( t_i \) without encountering other particles, and \( \delta_i = t_i - t_{i-1} \) denotes the distance between adjacent samples.

NeRFs [MST*20, NG21, BMT*21, BMV*22, VHM*22, LWC*23] have seen widespread success in problems such as editing [MBRS*21, ZLDD21, CZL*22, YSL*22], joint optimization of cameras [LMTL21, WWX*21, CCW*23, TRMT23], inverse rendering [ZLW*21, SDZ*21, BBJ*21, ZSD*21, ZZW*23, LZF*23], generalization [YYTK21, WWG*22, JFSL22, HZF*23b], acceleration [RPLG21, GJK*21, ZZZ*23b], and free-viewpoint video [DZY*21, LSN*22, PCPMNN21]. Apart from the above applications, NeRF-based representation can also be used for digit avatar generation, such as face and body reenactment [PDW*21, GCL*21, LHR*21, WCS*22, HPX*22]. NeRFs have been extend to various fields such as robotics [KFH*22, ZKW*23, ACC*22], tomography [RLW*22, ZLZ*22], image processing [HZF*22, MIL*22b, HZF*23a], and astronomy [LSC*22].

3.2.2. Neural Implicit Surfaces

Within the scope of shape reconstruction, a neural network processes a 3D coordinate as input and generates a scalar value, which usually signifies the signed distance to the surface. This method is particularly effective in filling in missing information and generating smooth, continuous surfaces. The implicit surface representation defines the scene’s surface as a learnable function \( f \) that specifies the signed distance \( f(x) \) from each point to the surface. The fundamental surface can then be extracted from the zero-level set, \( S = \{ x \in \mathbb{R}^3 | f(x) = 0 \} \), providing a flexible and efficient way to reconstruct complex 3D shapes. Implicit surface representations offer numerous advantages, as they eliminate the need to define mesh templates. As a result, they can represent objects with unknown or changing topology in dynamic scenarios. Specifically, implicit surface representations recover signed distance fields for shape modeling using MLPs with coordinate inputs. These initial proposals sparked widespread enthusiasm and led to various improvements focusing on different aspects, such as enhancing training schemes [DZW*20, YAK*20, ZML*22], leveraging global-local context [XWC*19, EGO*20, ZPL*22], adopting specific parame-
terizations [GCV \textsuperscript{19}, CTZ \textsuperscript{20}, YRSh \textsuperscript{21}, BSKG \textsuperscript{22}], and employing spatial partitions [GCS \textsuperscript{19}, TTG \textsuperscript{20}, TLY \textsuperscript{21}, WLG \textsuperscript{18}, WLY \textsuperscript{19}].

NeuS [WLL \textsuperscript{21}] and VoISDF [YKL \textsuperscript{21}] extend the basic NeRF formulation by integrating an SDF into volume rendering, which defines a function to map the signed distance to density $\alpha$. It attains a locally maximal value at surface intersection points. Specifically, accumulated transmittance $T(i)$ along the ray $r(i) = o + td$ is formulated as a sigmoid function: $T(i) = \mathbf{b}(f(i)) = (1 + e^{f(i)})^{-1}$, where $s$ and $f(i)$ refer to a learnable parameter and the signed distance function of points at $r(i)$, respectively. Discrete opacity values $\alpha_t$ can then be derived as:

$$\alpha_t = \max\left(\frac{\Phi_s(f(t_i)) - \Phi_s(f(t_{i+1}))}{\Phi_s(f(t_i))}, 0\right). \quad (2)$$

NeuS employs volume rendering to recover the underlying SDF based on Eqs. (1) and (2). The SDF is optimized by minimizing the photometric loss between the rendering results and ground-truth images.

Building upon NeuS and VoISDF, NeuralWarp [DBD \textsuperscript{22}], GeoNeuS [FOT \textsuperscript{22}], MonoSDF [YPN \textsuperscript{22}] leverage prior geometry information from MVS methods. IRON [ZLS \textsuperscript{22}], MII [ZSH \textsuperscript{22}], and WildLight [CLC \textsuperscript{23}] apply high-fidelity shape reconstruction via SDF for inverse rendering. HF-NeuS [WSW \textsuperscript{22}] and PET-Neus [WSW \textsuperscript{23}] integrate additional displacement networks to fit the high-frequency details. LoD-NeuS [ZZ \textsuperscript{23}] adaptively encodes Level of Detail (LoD) features for shape reconstruction.

3.3. Hybrid Representations

Implicit representations have indeed demonstrated impressive results in various applications as mentioned above. However, most of the current implicit methods rely on regression to NeRF or SDF values, which may limit their ability to benefit from explicit supervision on the target views or surfaces. Explicit representation could impose useful constraints during training and improve the user experience. To capitalize on the complementary benefits of both representations, researchers have begun exploring hybrid representations. These involve scene representations (either explicit or implicit) that embed features utilizing rendering algorithms for view synthesis.

3.3.1. Voxel Grids

Early work [WSK \textsuperscript{15}, CXG \textsuperscript{16}, MS15] depicted 3D shapes using voxels, which store coarse occupancy (inside/outside) values on a regular grid. This approach enabled powerful convolutional neural networks to operate natively and produce impressive results in 3D reconstruction and synthesis [DRB \textsuperscript{18}, WZX \textsuperscript{16}, BLW \textsuperscript{16}]. These methods usually use explicit voxel grids as the 3D representation. Recently, to address the slow training and rendering speeds of implicit representations, the 3D voxel-based embedding methods [LGZ \textsuperscript{20}, FKY \textsuperscript{22}, SSN \textsuperscript{22}, SSC \textsuperscript{22}] have been proposed. These methods encode the spatial information of the scene and decode the features more efficiently. Moreover, Instant-NGP [ME \textsuperscript{22}] introduces the multi-level voxel grids encoded implicitly via the hash function for each level. It facilitates rapid optimization and rendering while maintaining a compact model. These advancements in 3D shape representation and processing techniques have significantly enhanced the efficiency and effectiveness of 3D generation applications.

3.3.2. Tri-plane

Tri-plane representation is an alternative approach to using voxel grids for embedding features in 3D shape representation and neural rendering. The main idea behind this method is to decompose a 3D volume into three orthogonal planes (e.g., XY, XZ, and YZ planes) and represent the features of the 3D shape on these planes. Specifically, TensoRF [CXG \textsuperscript{22}] achieves similar model compression and acceleration by replacing each voxel grid with a tensor decomposition into planes and vectors. Tri-planes are efficient and capable of scaling with the surface area rather than volume and naturally integrate with expressive, fine-tuned 2D generative architectures. In the generative setting, EG3D [CLC \textsuperscript{22}] proposes a spatial decomposition into three planes whose values are added together to represent a 3D volume. NFD [SCP \textsuperscript{23}] introduces diffusion on 3D scenes, utilizing 2D diffusion model backbones and having built-in tri-plane representation.

3.3.3. Hybrid Surface Representation

DMTet, a recent development cited in [SG \textsuperscript{21}], is a hybrid three-dimensional surface representation that combines both explicit and implicit forms to create a versatile and efficient model. It segments the 3D space into dense tetrahedra, thereby forming an explicit partition. By integrating explicit and implicit representations, DMTet can be optimized more efficiently and transformed seamlessly into explicit structures like mesh representations. During the generation process, DMTet can be differentiably converted into a mesh, which enables swift high-resolution multi-view rendering. This innovative approach offers significant improvements in terms of efficiency and versatility in 3D modeling and rendering.

4. Generation Methods

In the past few years, the rapid development of generative models in 2D image synthesis, such as generative adversarial networks (GANs) [GPAM \textsuperscript{14}, AQ \textsuperscript{19}], variational autoencoders (VAEs) [KPHL \textsuperscript{17}, PGH \textsuperscript{16}, KW \textsuperscript{13}], autoregressive models [RC \textsuperscript{19}, BM \textsuperscript{20}], diffusion models [HJA \textsuperscript{20}, ND \textsuperscript{21}, SC \textsuperscript{22}], etc., has led to their extension and combination with these scene representations for 3D generation. Tab. 1 shows well-known examples of 3D generation using generative models and scene representations. These methods may use different scene representations in the generation space, where the representation is generated by the generative models, and the reconstruction space, where the output is represented. For example, AutoSDF [MC \textsuperscript{22}] uses a transformer-based autoregressive model to learn a feature voxel grid and decode this representation to SDF for reconstruction. EG3D [CLC \textsuperscript{22}] employs GANs to generate samples in latent space and introduces a tri-plane representation for rendering the output. SSDNeRF [CGC \textsuperscript{23}] uses the diffusion model to generate tri-plane features and decode them to NeRF for rendering. By leveraging the advantages of neural scene representations and generative models, these approaches have demonstrated remarkable potential in generating realistic and intricate 3D models while maintaining view consistency.
Table 1: Some examples of 3D generation methods. We first divide the methods according to the generative models and their corresponding representations in generation space. The representations in the reconstruction space determine how the 3D objects are formatted and rendered. We also list the main supervision and conditions of these methods. For the 2D supervision, a rendering technique is utilized to generate the images.

<table>
<thead>
<tr>
<th>Method</th>
<th>Generative Model</th>
<th>Generation Space</th>
<th>Reconstruction Space</th>
<th>Rendering</th>
<th>Supervision</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolyGen [NGEB20a]</td>
<td>Autoregressive</td>
<td>Polygon</td>
<td>Mesh</td>
<td>-</td>
<td>3D</td>
<td>Uncon./Label/Image</td>
</tr>
<tr>
<td>PointGrow [SWL*20a]</td>
<td>Autoregressive</td>
<td>Point</td>
<td>Point Cloud</td>
<td>-</td>
<td>3D</td>
<td>Uncon./Label/Image</td>
</tr>
<tr>
<td>gDNA [CJS*22]</td>
<td>GAN</td>
<td>Latent Code</td>
<td>Occupancy Field</td>
<td>Surface Rendering</td>
<td>2D&amp;3D</td>
<td>Uncon.</td>
</tr>
<tr>
<td>HoloDiffusion [KVNM23]</td>
<td>Diffusion</td>
<td>Voxel</td>
<td>NeRF</td>
<td>Volume Rendering</td>
<td>2D</td>
<td>Image</td>
</tr>
<tr>
<td>SSDNeRF [CGC*23]</td>
<td>Diffusion</td>
<td>Tri-plane</td>
<td>NeRF</td>
<td>Volume Rendering</td>
<td>2D</td>
<td>Uncon./Image</td>
</tr>
<tr>
<td>3DShape2VecSet [ZTNW23]</td>
<td>Diffusion</td>
<td>Latent Set</td>
<td>SDF</td>
<td>-</td>
<td>3D</td>
<td>Uncon./Text/Image</td>
</tr>
<tr>
<td>Point-E [NJD*22]</td>
<td>Diffusion</td>
<td>Point</td>
<td>Point Cloud</td>
<td>-</td>
<td>3D</td>
<td>Text</td>
</tr>
<tr>
<td>3DGen [GXX*23]</td>
<td>Diffusion</td>
<td>Tri-plane</td>
<td>Mesh</td>
<td>-</td>
<td>3D</td>
<td>Text/Image</td>
</tr>
<tr>
<td>DreamFusion [PJBM23]</td>
<td>Diffusion</td>
<td>-</td>
<td>NeRF</td>
<td>Volume Rendering</td>
<td>SDS</td>
<td>Text</td>
</tr>
<tr>
<td>Make-It-3D [TWZ*23]</td>
<td>Diffusion</td>
<td>-</td>
<td>Point Cloud</td>
<td>Network Rendering</td>
<td>SDS</td>
<td>Image</td>
</tr>
<tr>
<td>Zero-1-to-3 [LWVH*23]</td>
<td>Diffusion</td>
<td>Pixel</td>
<td>-</td>
<td>-</td>
<td>2D</td>
<td>Image</td>
</tr>
<tr>
<td>MVDream [SWY*23]</td>
<td>Diffusion</td>
<td>Pixel</td>
<td>-</td>
<td>-</td>
<td>2D</td>
<td>Image</td>
</tr>
<tr>
<td>DMV3D [XTL*23]</td>
<td>Diffusion</td>
<td>Tri-plane</td>
<td>Volume Rendering</td>
<td>2D</td>
<td>Text/Image</td>
<td></td>
</tr>
</tbody>
</table>

In this section, we explore a large variety of 3D generation methods which are organized into four categories based on their algorithmic paradigms: Feedforward Generation (Sec. 4.1), generating results in a forward pass; Optimization-Based Generation (Sec. 4.2), necessitating a test-time optimization for each generation; Procedural Generation (Sec. 4.3), creating 3D models from sets of rules; and Generative Novel View Synthesis (Sec. 4.4), synthesizing multi-view images rather than an explicit 3D representation for 3D generation. An evolutionary tree of 3D generation methods is depicted in Fig. 4, which illustrates the primary branch of generation techniques, along with associated work and subsequent developments. A comprehensive analysis will be discussed in the subsequent subsection.

### 4.1. Feedforward Generation

A primary technical approach for generation methods is feedforward generation, which can directly produce 3D representations using generative models. In this section, we explore these methods based on their generative models as shown in Fig. 5, which include generative adversarial networks (GANs), diffusion Models, autoregressive models, variational autoencoders (VAEs) and normalizing flows.

#### 4.1.1. Generative Adversarial Networks

Generative Adversarial Networks (GANs) [GPAM*14] have demonstrated remarkable outcomes in image synthesis tasks, consisting of a generator \( G(\cdot) \) and a discriminator \( D(\cdot) \). The generator network \( G \) produces synthetic data by accepting latent code as input, while the discriminator network \( D \) differentiates between generated data from \( G \) and real data. Throughout the training optimization process, the generator \( G \) and discriminator \( D \) are jointly optimized, guiding the generator to create synthetic data as realistic as real data.

Building on the impressive results achieved by GANs in 2D image synthesis, researchers have begun to explore the application of these models to 3D generation tasks. The core idea is to marry GANs with various 3D representations, such as point clouds (l-GAN/r-GAN [ADMG18], tree-GAN [SPK19]), voxel grids (3D-GAN [WZX*16], Z-GAN [KKR18]), meshes (MeshGAN [CBZ*19]), or SDF (SurfGen [LLZL21], SDF-StyleGAN [ZLWT22]). In this context, the 3D generation process can be viewed as a series of adversarial steps, where the generator learns to create realistic 3D data from input latent codes, and the discriminator differentiates between generated data and real data. By iteratively optimizing the generator and discriminator networks, GANs learn to generate 3D data that closely resembles the realism of actual data.

For 3D object generation, prior GAN methodologies, such as l-GAN [ADMG18], 3D-GAN [WZX*16], and Multi-chart Generation [BHMK*18], directly utilize explicit 3D object representation of real data to instruct generator networks. Their discriminators employ 3D representation as supervision, directing the gener-
Figure 4: The evolutionary tree of 3D generation illustrates the primary branch of generation methods and their developments in recent years. Specifically, we provide a comprehensive overview of the rapidly growing literature on generation methods, categorized by the type of algorithmic paradigms, including feedforward generation, optimization-based generation, procedural generation, and generative novel view synthesis.
Figure 5: Exemplary feedforward 3D generation models. We showcase several representative pipelines of feedforward 3D generation models, including (a) generative adversarial networks, (b) diffusion models, (c) autoregressive models, (d) variational autoencoders and (e) normalizing flows.

ator to produce synthetic data that closely resembles the realism of actual data. During training, specialized generators generate corresponding supervisory 3D representations, such as point clouds, voxel grids, and meshes. Some studies, like SurfGen [LLZL21], have progressed further to generate intermediate implicit representations and then convert them to corresponding 3D representations instead of directly generating explicit ones, achieving superior performance. In particular, the generator of l-GAN [ADMG18], 3D-GAN [WZX*16], and Multi-chart Generation [BHMK*18] generate the position of point cloud, voxel grid, and mesh directly, respectively, taking latent code as input. SurfGen [LLZL21] generates implicit representation and then extracts explicit 3D representation.

In addition to GANs that directly generate various 3D representations, researchers have suggested incorporating 2D supervision through differentiable rendering to guide 3D generation, which is commonly referred to as 3D-Aware GAN. Given the abundance of 2D images, GANs can better understand the implicit relationship between 2D and 3D data than relying solely on 3D supervision. In this approach, the generator of GANs generates rendered 2D images from implicit or explicit 3D representation. Then the discriminators distinguish between rendered 2D images and real 2D images to guide the training of the generator.

Specifically, HoloGAN [NPLT*19] first learns a 3D representation of 3D features, which is then projected to 2D features by the camera pose. These 2D feature maps are then rendered to generate the final images. BlockGAN [NPRM*20] extends it to generate 3D features of both background and foreground objects and combine them into 3D features for the whole scene. In addition, PrGAN [GMW17] and PlatonicGAN [HMR19a] employ an explicit voxel grid structure to represent 3D shapes and use a render layer to create images. Other methods like DIB-R [CZ19], ConvMesh [PSH*20], Textured3DGAN [PKHL21] and GET3D [GSW*22] propose GAN frameworks for generating triangle meshes and textures using only 2D supervision.

Building upon representations such as NeRFs, GRAF [SLNG20] proposes generative radiance fields utilizing adversarial frameworks and achieves controllable image synthesis at high resolutions. pi-GAN [CMK*21a] introduces SIREN-based implicit GANs with FiLM conditioning to further improve image quality and view consistency. GIRAFFE [NG21] represents scenes as compositional generative neural feature fields to model multi-object scenes. Furthermore, EG3D [CLC*22] first proposes a hybrid explicit–implicit tri-plane representation that is both efficient and expressive and has been widely adopted in many following works.

4.1.2. Diffusion Models

Diffusion models [HJA20,RBL*22a] are a class of generative models that learn to generate data samples by simulating a diffusion process. The key idea behind diffusion models is to transform the original data distribution into a simpler distribution, such as Gaussian, through a series of noise-driven steps called the forward process. The model then learns to reverse this process, known as the backward process, to generate new samples that resemble the original data distribution. The forward process can be thought of as gradually adding noise to the original data until it reaches the target distribution. The backward process, on the other hand, involves iteratively denoising the samples from the distribution to generate the final output. By learning this denoising process, diffusion models can effectively capture the underlying structure and patterns of the data, allowing them to generate high-quality and diverse samples.

Building on the impressive results achieved by diffusion models in generating 2D images, researchers have begun to explore the applications of these models to 3D generation tasks. The core idea is to marry denoising diffusion models with various 3D representations. In this context, the 3D generation process can be viewed as a series of denoising steps, reversing the diffusion process from input 3D data to Gaussian noise. The diffusion models learn to generate 3D data from this noisy distribution through denoising.
Specifically, Cai et al. [CYAE*20] build upon a denoising score-matching framework to learn distributions for point cloud generation. PVD [ZDW21] combines the benefits of both point-based and voxel-based representations for 3D generation. The model learns a diffusion process that transforms point clouds into voxel grids and vice versa, effectively capturing the underlying structure and patterns of the 3D data. Similarly, DPM [LH21] focuses on learning a denoising process for point cloud data by iterative denoising the noisy point cloud samples. Following the advancements made by PVD [ZDW21] and DPM [LH21], LION [ZVW*22] builds upon the idea of denoising point clouds and introduces the concept of denoising in the latent space of point clouds, which is analogous to the shift in 2D image generation from denoising pixels to denoising latent space representations. To generate point clouds from text prompts, Point-E [NJD*22] initially employs the GLIDE model [NDR*21] to generate text-conditional synthetic views, followed by the production of a point cloud using a diffusion model conditioned on the generated image. By training the model on a large-scale 3D dataset, it achieves remarkable generalization capabilities.

In addition to point clouds, MeshDiffusion [LFB*23], Tetrahedral Diffusion Models [KPWS22], and SLIDE [LWA*23] explore the application of diffusion models to mesh generation. MeshDiffusion [LFB*23] adopts the DMTet representation [SGY*21] for meshes and optimizes the model by treating the optimization of signed distance functions as a denoising process. Tetrahedral Diffusion Models [KPWS22] extend diffusion models to tetrahedral meshes, learning displacement vectors and signed distance values on the tetrahedral grid through denoising. SLIDE [LWA*23] explores diffusion models on sparse latent points for mesh generation.

Apart from applying diffusion operations on explicit 3D representations, some works focus on performing the diffusion process on implicit representations. SSDNeRF [CGC*23], DiffRF [MSP*23] and Shap-E [JN23] operate on 3D radiance fields, while SDF-Diffusion [SKJ23], LAS-Diffusion [ZPW*23], Neural Wavelet-domain Diffusion [HLHF22], One-2-3-45++ [LXJ*23], SD Fusion [CLT*23] and 3D-LDM [NKRR*22] focus on signed distance fields representations. Specifically, Diffusion-SDF [LDZL23] utilizes a voxel-shaped SDF representation to generate high-quality and continuous 3D shapes. 3D-LDM [NKRR*22] creates neural implicit representations of SDFs by initially using a diffusion model to generate the latent space of an auto-decoder. Subsequently, the latent space is decoded into SDFs to acquire 3D shapes. Moreover, Rodin [WZZ*23] and Shue et al. [SCP*23] adopt tri-plane as the representation and optimize the tri-plane features using diffusion methods. Shue et al. [SCP*23] generates 3D shapes using occupancy networks, while Rodin [WZZ*23] obtains 3D shapes through volumetric rendering.

These approaches showcase the versatility of diffusion models in managing various 3D representations, including both explicit and implicit forms. By tailoring the denoising process to different representation types, diffusion models can effectively capture the underlying structure and patterns of 3D data, leading to improved generation quality and diversity. As research in this area continues to advance, it is expected that diffusion models will play a crucial role in pushing the boundaries of 3D shape generation across a wide range of applications.

4.1.3. Autoregressive Models

A 3D object can be represented as a joint probability of the occurrences of multiple 3D elements:

\[
p(x_0, x_1, ..., x_n),
\]

where \(x_i\) is the \(i\)-th element which can be the coordinate of a point or a voxel. A joint probability with a large number of random variables is usually hard to learn and estimate. However, one can factorize it into a product of conditional probabilities:

\[
p(x_0, x_1, ..., x_n) = p(x_0) \prod_{i=1}^n p(x_i|x_{<i}),
\]

which enables learning conditional probabilities and estimating the joint probability via sampling. Autoregressive models for data generation are a type of models that specify the current output depending on their previous outputs. Assuming that the elements \(x_0, x_1, ..., x_n\) form an ordered sequence, a model can be trained by providing it with previous inputs \(x_0, ..., x_{i-1}\) and supervising it to fit the probability of the outcome \(x_i\):

\[
p(x_i|x_{<i}) = f(x_0, ..., x_{i-1}),
\]

the conditional probabilities are learned by the model function \(f\). This training process is often called teacher forcing. The model can be then used to autoregressively generate the elements step-by-step:

\[
x_i = \arg\max_j p(x|x_{<i}).
\]

State-of-the-art generative models such as GPTs [RWC*19, BMR*20] are autoregressive generators with Transformer networks as the model function. They achieve great success in generating natural languages and images. In 3D generation, several studies have been conducted based on autoregressive models. In this section, we discuss some notable examples of employing autoregressive models for 3D generation.

PointGrow [SWL*20b] generates point clouds using an autoregressive network with self-attention context awareness operations in a point-by-point manner. Given its previously generated points, PointGrow refolds the points by axes and passes them into three branches. Each branch takes the inputs to predict a coordinate value of one axis. The model can also condition an embedding vector to generate point clouds, which can be a class category or an image. Inspired by the network from PointGrow, PolyGen [NGEB20b] generates 3D meshes with two transformer-based networks, one for vertices and one for faces. The vertex transformer autoregressively generates the next vertex coordinate based on previous vertices. The face transformer takes all the output vertices as context to generate faces. PolyGen can condition on a context of object classes or images, which are cross-attended by the transformer networks.

Recently, AutoSDF [MCST22b] generates 3D shapes represented by volumetric truncated-signed distance function (T-SDF). AutoSDF learns a quantized codebook regarding local regions of T-SDFs using VQ-VAE. The shapes are then presented by the codebook tokens and learned by a transformer-based network in a non-sequential autoregressive manner. In detail, given previous
tokens at arbitrary locations and a query location, the network predicts the token that is queried. AutoSDF is capable of completing shapes and generating shapes based on images or text. Concurrently with AutoSDF, ShapeFormer [YLM*22] generates surfaces of 3D shapes based on incomplete and noisy point clouds. A compact 3D representation called vector quantized deep implicit function (VQDIF) is used to represent shapes using a feature sequence of discrete variables. ShapeFormer first encodes an input point cloud into a partial feature sequence. It then uses a transformer-based network to autoregressively sample out the complete sequence. Finally, it decodes the sequence to a deep implicit function from which the complete object surface can be extracted. Instead of learning in 3D volumetric space, Luo et al. propose an improved auto-regressive model (ImAM) to learn discrete representation in a one-dimensional space to enhance the efficient learning of 3D shape generation. The method first encodes 3D shapes of volumetric grids into three axis-aligned planes. It uses a coupling network to further project the planes into a latent vector, where vector quantization is performed for discrete tokens. ImAM adopts a vanilla transformer to autoregressively learn the tokens with tractable orders. The generated tokens are decoded to occupancy values via a network by sampling spatial locations. ImAM can switch from unconditional generation to conditional generation by concatenating various conditions, such as point clouds, categories, and images.

4.1.4. Variational Autoencoders

Variational autoencoders (VAEs) [KW13] are probabilistic generative models that consist of two neural network components: the encoder and decoder. The encoder maps the input data point to a latent space that corresponds to the parameters of a variational distribution. In this way, the encoder can produce multiple different samples that all come from the same distribution. The decoder maps from the latent space to the input space, to produce or generate data points. Both networks are typically trained together with the usage of the reparameterization trick, although the variance of the noise model can be learned separately. VAEs have also been explored in 3D generation [KYLH21,GYW*21,GYW*19b,BLW16,KSZ*21].

Brock et al. trains variational autoencoders directly for voxels using 3D ConvNet, while SDM-Net [GYW*19b] focuses on the generation of structured meshes composed of deformable parts. The method uses one VAE network to model parts and another to model the whole object. The follow-up work TMNet [GYW*21] could generate texture maps of meshes in a part-aware manner. Other representations like point clouds [KYLH21] and NeRFs [KSZ*21] are also explored in variational autoencoders. Owing to the reconstruction-focused objective of VAEs, their training is considerably more stable than that of GANs. However, VAEs tend to produce more blurred results compared to GANs.

4.1.5. Normalizing Flows

Normalizing flow models consist of a series of invertible transformations that map a simple distribution, such as Gaussian, to a target distribution, which represents the data to generate. These transformations are carefully designed to be differentiable and invertible, allowing one to compute the likelihood of the data under the model and optimize the model parameters using gradient-based optimization techniques.

In 3D generation, PointFlow [YHH*19a] learns a distribution of shapes and a distribution of points using continuous normalizing flows. This approach allows for the sampling of shapes, followed by the sampling of an arbitrary number of points from a given shape. Discrete PointFlow (DPF) network [KBV20] improves PointFlow by replacing continuous normalizing flows with discrete normalizing flows, which reduces the training and sampling time. SoftFlow [KLK*20] is a framework for training normalizing flows on the manifold. It estimates a conditional distribution of the perturbed input data instead of learning the data distribution directly. SoftFlow alleviates the difficulty of forming thin structures for flow-based models.

4.2. Optimization-based Generation

Optimization-based generation is employed to generate 3D models using runtime optimization. These methods usually leverage pre-trained multimodal networks to optimize 3D models based on user-specified prompts. The key lies in achieving alignment between the given prompts and the generated content while maintaining high fidelity and diversity. In this section, we primarily examine optimization-based generation methods that use texts and images, based on the types of prompts provided by users.

4.2.1. Text-to-3D

Language serves as the primary means of human communication and describing scenes, and researchers are dedicated to exploring the potential of text-based generation methods. These methods typically align the text with the images obtained through the differentiable rendering techniques, thereby guiding the generation of 3D content based on the text prompts. Given a fixed surface, TANGO [LZJ*22] uses CLIP [RKH*21a] to supervise differentiable physical-based rendering (PBR) images and obtain texture maps that align with the specified text prompt. Inspired by the success of NeRF [MST*20] and diffusion models in modeling 3D static scenes and text-to-image tasks respectively, DreamFusion [PJBM23] (as shown in Fig. 6) combines the volumetric...
Table 2: Quantitative comparison of image-to-3D methods on surface reconstruction. We summarize the Chamfer distance and volume IoU as the metrics to evaluate the quality of surface reconstruction.

<table>
<thead>
<tr>
<th>Method</th>
<th>Chamfer Distance ↓</th>
<th>Volume IoU ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealFusion [MKLRV23]</td>
<td>0.0819</td>
<td>0.2741</td>
</tr>
<tr>
<td>Magic123 [QMH+23]</td>
<td>0.0516</td>
<td>0.4528</td>
</tr>
<tr>
<td>Make-it-3D [TWZ+23]</td>
<td>0.0732</td>
<td>0.2937</td>
</tr>
<tr>
<td>One-2-3-4-S [LXJ+23]</td>
<td>0.0629</td>
<td>0.4086</td>
</tr>
<tr>
<td>Point-E [NJD+22]</td>
<td>0.0426</td>
<td>0.2875</td>
</tr>
<tr>
<td>Shap-E [JN23]</td>
<td>0.0436</td>
<td>0.3584</td>
</tr>
<tr>
<td>Zero-1-to-3 [LWVH+23]</td>
<td>0.0339</td>
<td>0.5035</td>
</tr>
<tr>
<td>SyncDreamer [LLZ+23]</td>
<td>0.0261</td>
<td>0.5421</td>
</tr>
</tbody>
</table>

Table 3: Quantitative comparison of image-to-3D methods on novel view synthesis. We report the CLIP-Similarity, PSNR, and LPIPS as the metrics to evaluate the quality of view synthesis.

<table>
<thead>
<tr>
<th>Method</th>
<th>CLIP-Similarity ↑</th>
<th>PSNR ↑</th>
<th>LPIPS ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealFusion [MKLRV23]</td>
<td>0.735</td>
<td>20.216</td>
<td>0.197</td>
</tr>
<tr>
<td>Magic123 [QMH+23]</td>
<td>0.747</td>
<td>25.637</td>
<td>0.062</td>
</tr>
<tr>
<td>Make-it-3D [TWZ+23]</td>
<td>0.839</td>
<td>20.010</td>
<td>0.119</td>
</tr>
<tr>
<td>One-2-3-4-S [LXJ+23]</td>
<td>0.788</td>
<td>23.159</td>
<td>0.096</td>
</tr>
<tr>
<td>Zero-1-to-3 [LWVH+23]</td>
<td>0.759</td>
<td>25.386</td>
<td>0.068</td>
</tr>
<tr>
<td>SyncDreamer [LLZ+23]</td>
<td>0.837</td>
<td>25.896</td>
<td>0.059</td>
</tr>
</tbody>
</table>

representation used in NeRF with the proposed Score Distillation Sampling (SDS) loss to achieve high-fidelity 3D content generation. SDS loss converts rendering error minimization into probability density estimation and enables 2D diffusion priors to optimize 3D representations (e.g., volumetric representation and triangle mesh) via image parameterization (e.g., differentiable rendering). As a concurrent work concurrent with SDS, Score Jacobian Chaining (SJC) [WDL+23] interprets predictions from pre-trained diffusion models as a score function of the data log-likelihood, similarly enabling 2D diffusion priors to optimize 3D representations via score matching. Based on DreamFusion, Magic3D [LGT+23] introduces a coarse-to-fine manner and extracts the underlying geometry of the volume as a mesh. It then combines differentiable neural rendering and SDS to refine the extracted mesh. Magic3D is capable of exporting high-quality textured meshes and seamlessly embedding them into the traditional graphics pipeline. Also as a two-stage method, Fantasia3D further combines DMTet [SGY+21] and SDS in the first geometry modeling stage to explicitly optimize surface. In the second stage, it introduces the PBR material model and disentangles texture and environment illumination. ProlificDreamer [WLW+23] presents variational score distillation (VSD) to boost text-to-3D generation. VSD adopts particles to model the distribution of 3D scenes and derive a gradient-based optimization scheme from the Wasserstein gradient flow, narrowing the gap between the rendering results distribution of the modeling distribution and pre-trained diffusion distribution. Benefiting from the optimization of scene distribution rather than a single scene, VSD overcomes the over-smoothed and over-saturated results produced by SDS and improves diversities. MVDream [SWY+23] further fine-tunes a multi-view diffusion model and introduces multi-view consistent 3D priors, overcoming multi-face and content-drift problems. Text-to-3D has gained significant attention recently, in addition to these, many other methods [ZZ23, LCCT23, MRP+23a] have been proposed in this field.

4.2.2. Image-to-3D

As the primary way to describe the visual effects of scenes, images can more intuitively describe the details and appearance of scenes at a finer-grained than language. Recent works thus are motivated to explore the image-to-3D techniques, which reconstruct remarkable and high-fidelity 3D models from specified images. These methods strive to maintain the appearance of the specified images and optimized 3D contents while introducing reasonable geometric priors. Similar to the text-to-3D methods, several image-to-3D methods leverage the volumetric representation used in NeRF to represent the target 3D scenes, which natively introduces multi-view consistency. NeuralLift-360 [XJW+22] uses estimated monocular depth and CLIP-guided diffusion prior to regularizing the geometry and appearance optimization respectively, achieving lift of a single image to a 3D scene represented by a NeRF. RealFusion [MKLRV23] and NeRDi [DJQ+23] leverage textual inversion [GAA+22] to extract text embeddings to condition a pre-trained image diffusion model [RBL+22b], and combine use the score distillation loss to optimize the volumetric representation. Based on Magic3D [LGT+23] that employs a coarse-to-fine framework as mentioned above, Magic123 [QMH+23] additionally introduces 3D priors from a pre-trained viewpoint-conditioned diffusion model Zero-1-to-3 [LWVH+23] in two optimization stage, yielding textured meshes that match the specified images. As another two-stage image-to-3D method, Make-it-3D [TWZ+23] enhances texture and geometric structure in the fine stage, producing high-quality textured point clouds as final results. Subsequent works [SZS+23, YYY+23] have been consistently proposed to enhance the previous results. Recently, 3D Gaussian Splatting (3DGS) [KKL+23] has emerged as a promising modeling as well as a real-time rendering technique. Based on 3DGS, DreamGaussian [TRZ+23] presents an efficient two-stage framework for both text-driven and image-driven 3D generation. In the first stage, DreamGaussian leverages SDS loss (i.e., 2D diffusion priors [LWVH+23] and CLIP-guided diffusion priors [PJBM23]) to generate target objects represented by 3D Gaussians. Then DreamGaussian extracts textured mesh from the optimized 3D Gaussians by querying the local density and refines textures in the UV space. For a better understanding of readers to various image-to-3D methods, we evaluate the performance of some open-source state-of-the-art methods. Tab. 2 shows the quantitative comparison of image-to-3D methods on surface reconstruction. We summarize the Chamfer distance and volume IoU as the metrics to evaluate the quality of surface reconstruction. Tab. 3 demonstrates the quantitative comparison of image-to-3D methods on novel view synthesis. We report the CLIP-Similarity, PSNR, and LPIPS as the metrics to evaluate the quality of view synthesis.
4.3. Procedural Generation

Procedural generation is a term for techniques that create 3D models and textures from sets of rules. These techniques often rely on predefined rules, parameters, and mathematical functions to generate diverse and complex content, such as textures, terrains, levels, characters, and objects. One of the key advantages of procedural generation is their ability to efficiently create various shapes from a relatively small set of rules. In this section, we mainly survey four most used techniques: fractal geometry, L-Systems, noise functions and cellular automata.

A fractal [Man67, MM82] is a geometric shape that exhibits detailed structure at arbitrarily small scales. A characteristic feature of many fractals is their similarity across different scales. This property of exhibiting recurring patterns at progressively smaller scales is referred to as self-similarity. A common application of fractal geometry is the creation of landscapes or surfaces. These are generated using a stochastic algorithm designed to produce fractal behavior that mimics the appearance of natural terrain. The resulting surface is not deterministic, but rather a random surface that exhibits fractal behavior.

An L-system [Lin68], or Lindenmayer system, is a type of formal grammar and parallel rewriting system. It comprises an alphabet of symbols that can be utilized to construct strings, a set of production rules that transform each symbol into a more complex string of symbols, a starting string for construction, and a mechanism for converting the produced strings into geometric structures. L-systems are used to create complex and realistic 3D models of natural objects like trees and plants. The string generated by the L-system can be interpreted as instructions for a “turtle” to move in 3D space. For example, certain characters might instruct the turtle to move forward, turn left or right, or push and pop positions and orientations onto a stack.

Noise functions, such as Perlin noise [Per85] and Simplex noise [Per02], are used to generate coherent random patterns that can be applied to create realistic textures and shapes in 3D objects. These functions can be combined and layered to create more complex patterns and are particularly useful in terrain generation, where they can be used to generate realistic landscapes with varying elevations, slopes, and features.

Cellular automata [VN*51, Neu66, Wol83] are a class of discrete computational models that consist of a grid of cells, each of which can be in one of a finite number of states. The state of each cell is determined by a set of rules based on the states of its neighboring cells. Cellular automata have been used in procedural generation to create various 3D objects and patterns, such as cave systems, mazes, and other structures with emergent properties.

4.4. Generative Novel View Synthesis

Recently, generative techniques have been utilized to tackle the challenge of novel view synthesis, particularly in predicting new views from a single input image. Compared to the conventional 3D generation methods, it does not explicitly utilize the 3D representation to enforce 3D consistency, instead, it usually employs a 3D-aware method by conditioning 3D information. In the field of novel view synthesis, a widely studied technical route will be regression-based methods [YYTK21, WWG*21, CXZ*21]. Different from them, generative novel view synthesis focuses more on generating new content rather than regressing the scenes from a few input images, which typically involves long-range view extrapolation.

With the development of image synthesis methods, significant progress has been made in generative novel view synthesis. Recently, 2D diffusion models have transformed image synthesis and therefore are also utilized in generative novel view synthesis [WCMB*22, TLK*23, LWVH*23, CNC*23, TYC*23, YGMG23]. Among these methods, 3DiM [WCMB*22] first introduces a geometry-free-image-to-image diffusion model for novel view synthesis, taking the camera pose as the condition. Tseng et al. [TLK*23] designs epipolar attention layers to inject camera parameters into the pose-guided diffusion model for consistent view synthesis from a single input image. Zero-1-to-3 [LWVH*23] (as shown in Fig. 7) demonstrates the learning of the camera viewpoint in large-scale diffusion models for zero-shot novel view synthesis. [CNC*23, TYC*23, YGMG23] condition 2D diffusion models on pixel-aligned features extracted from input views to extend them to be 3D-aware. However, generating multiview-consistent images remains a challenging problem. To ensure consistent generation, [LHG*23, LLZ*23, SCZ*23, LGL*23] propose a multi-view diffusion model that could synthesize multi-view images simultaneously to consider the information between different views, which achieve more consistent results compared to the single view synthesis model like Zero-1-to-3 [LWVH*23].

Prior to that, the transformer which is a sequence-to-sequence model originally proposed in natural language processing, uses a multi-head attention mechanism to gather information from different positions and brings lots of attention in the vision community. Many tasks achieve state-of-the-art performance using the attention mechanism from the transformer including generative novel view synthesis [REO21, SMP*22, KDSB22]. Specifically, Geometry-free View Synthesis [REO21] learns the discrete repre-
Some methods applied to novel view synthesis produce high-quality results in image synthesis and consequently are implicit representation. ViewFormer employs an encoder-decoder model based on transformers to learn an (VQ-VAE) codebook and a transformer model. While ViewFormer also uses a two-stage training consisting of a Vector Quantized Variational Autoencoder (VQ-VAE) and transformers. While ViewFormer also uses a two-stage presentation vis VQGAN to obtain an abstract latent space for training transformers. While ViewFormer also uses a two-stage training consisting of a Vector Quantized Variational Autoencoder (VQ-VAE) and a transformer model. And ViewFormer employs an encoder-decoder model based on transformers to learn an implicit representation.

On the other hand, generative adversarial networks could produce high-quality results in image synthesis and consequently are applied to novel view synthesis. Some methods maintain a 3D point cloud as the representation, which could be projected onto novel views followed by a GAN to hallucinate the missing regions and synthesize the output image. While and focus on long-range view generation via transformers. At an earlier stage of deep learning methods when the auto-encoders and variational autoencoders begin to be explored, it is also used to synthesize the novel views.

In summary, generative novel view synthesis can be regarded as a subset of image synthesis techniques and continues to evolve alongside advancements in image synthesis methods. Besides the generative models typically included, determining how to integrate information from the input view as a condition for synthesizing the novel view is the primary issue these methods are concerned with.

5. Datasets for 3D Generation

With the rapid development of technology, the ways of data acquisition and storage become more feasible and affordable, resulting in an exponential increase in the amount of available data. As data accumulates, the paradigm for problem-solving gradually shifts from data-driven to model-driven approaches, which in turn contributes to the growth of "Big Data" and "AIGC". Nowadays, data plays a crucial role in ensuring the success of algorithms. A well-curated dataset can significantly enhance a model’s robustness and performance. On the contrary, noisy and flawed data may cause model bias that requires considerable effort in algorithm design to rectify.

In this section, we will go over the common data used for 3D generation. Depending on the methods employed, it usually includes 3D data (Section 5.1), multi-view image data (Section 5.2), and single-view image data (Section 5.3), which are also summarized in Tab. 4.

### 5.1. Learning from 3D Data

3D data could be collected by RGB-D sensors and other technology for scanning and reconstruction. Apart from 3D generation, 3D data is also widely used for other tasks like helping improve classical 2D vision task performance by data synthesis, environment simulation for training embodied AI agents, 3D object understanding, etc. One popular and frequently used 3D model database in the early stage is the Princeton Shape Benchmark. It contains about 1800 polygonal models collected from the World Wide Web. While constructs a special rig that contains a 3D digitizer, a turntable, and a pair of cameras mounted on a sled that can move along a bent rail to capture the kit object models database. To evaluate the algorithms to detect and estimate the objects in the image given 3D models, introduces a dataset of 3D IKEA models obtained from Google Warehouse. Some 3D model databases are presented for tasks like robotic manipulation, introduces a dataset of 3D IKEA models obtained from Google Warehouse. Some 3D model databases are presented for tasks like robotic manipulation, introduces a dataset of 3D IKEA models obtained from Google Warehouse. Some 3D model databases are presented for tasks like robotic manipulation, introduces a dataset of 3D IKEA models obtained from Google Warehouse. Some 3D model databases are presented for tasks like robotic manipulation, introduces a dataset of 3D IKEA models obtained from Google Warehouse. Some 3D model databases are presented for tasks like robotic manipulation, introduces a dataset of 3D IKEA models obtained from Google Warehouse. Some 3D model databases are presented for tasks like robotic manipulation, introduces a dataset of 3D IKEA models obtained from Google Warehouse. Some 3D model databases are presented for tasks like robotic manipulation.

### Table 4: Selected datasets commonly used for 3D generation.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Type</th>
<th>Year</th>
<th>Samples</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShapeNet [CFG+15]</td>
<td>3D data</td>
<td>2015</td>
<td>51K</td>
<td>objects</td>
</tr>
<tr>
<td>Thingi10K [ZJ16]</td>
<td>3D data</td>
<td>2016</td>
<td>10K</td>
<td>objects</td>
</tr>
<tr>
<td>3D-Future [FJG+21]</td>
<td>3D data</td>
<td>2020</td>
<td>10K</td>
<td>furniture</td>
</tr>
<tr>
<td>GSO [DFK+22]</td>
<td>3D Data</td>
<td>2022</td>
<td>1K</td>
<td>household items</td>
</tr>
<tr>
<td>Objaverse [DSS+23]</td>
<td>3D data</td>
<td>2022</td>
<td>800K</td>
<td>objects</td>
</tr>
<tr>
<td>OmniObject3D [WZF+23]</td>
<td>3D data</td>
<td>2023</td>
<td>6K</td>
<td>objects</td>
</tr>
<tr>
<td>Objaverse-XL [DLW+23]</td>
<td>3D Data</td>
<td>2023</td>
<td>10.2M</td>
<td>objects</td>
</tr>
<tr>
<td>ScanNet [DCS+17]</td>
<td>multi-view images</td>
<td>2017</td>
<td>1.5K</td>
<td>indoor scenes</td>
</tr>
<tr>
<td>CO3D [RSH+21]</td>
<td>multi-view images</td>
<td>2021</td>
<td>19K</td>
<td>objects</td>
</tr>
<tr>
<td>MVImgNet [YXZ+23]</td>
<td>multi-view images</td>
<td>2023</td>
<td>219K</td>
<td>indoor scenes</td>
</tr>
<tr>
<td>DeepFashion [LLQ+16]</td>
<td>single-view images</td>
<td>2016</td>
<td>800K</td>
<td>clothes</td>
</tr>
<tr>
<td>FFHQ [KLA19]</td>
<td>single-view images</td>
<td>2018</td>
<td>70K</td>
<td>human faces</td>
</tr>
<tr>
<td>AFHQ [CUYH20]</td>
<td>single-view images</td>
<td>2019</td>
<td>15K</td>
<td>animal faces</td>
</tr>
<tr>
<td>SHHQ [FLJ+22]</td>
<td>single-view images</td>
<td>2022</td>
<td>40K</td>
<td>human bodies</td>
</tr>
</tbody>
</table>

However, those datasets are very small and only contain hundreds or thousands of objects. Collecting, organizing, and labeling larger datasets in computer vision and graphics communities is needed for data-driven methods of 3D content. To address this, ShapeNet is introduced to build a large-scale repository of 3D CAD models of objects. The core of ShapeNet covers 55 common object categories with about 51,300 models that are manually verified category and alignment annotations. Thingi10K [ZJ16] collects 10,000 3D printing models from an online repository Thinger. While PhotoShape [PRFS18] produces 11,000 photorealistic...
tic, relightable 3D shapes based on online data. Other datasets such as 3D-Future [FJG’21], ABO [CGD’22], GSO [DFK’22] and OmniObject3D [WZF’23] try to improve the texture quality but only contain thousands of models. Recently, Objaverse [DSS’23] presents a large-scale corpus of 3D objects that contains over 800K 3D assets for research in the field of AI and makes a step toward presenting a large-scale 3D dataset. Objaverse-XL [DLW’23] further extends Objaverse to a larger 3D dataset of 10.2M unique objects from a diverse set of sources. These large-scale 3D datasets have the potential to facilitate large-scale training and boost the performance of 3D generation.

5.2. Learning from Multi-view Images

3D objects have been traditionally created through manual 3D modeling, object scanning, conversion of CAD models, or combinations of these techniques [DFK’22]. These techniques may only produce synthetic data or real-world data of specific objects with limited reconstruction accuracy. Therefore, some datasets directly provide multi-view images in the wild which are also widely used in many 3D generation methods. ScanNet [DCS’17] introduces an RGB-D video dataset containing 2.5M views in 1513 scenes and Objectron [AZA’21] contains object-centric short videos and includes 4 million images in 14,819 annotated videos, of which only a limited number cover the full 360 degrees. CO3D [RSH’21] extends the dataset from [HRL’21] and increases the size to nearly 19,000 videos capturing objects from 50 MS-COCO categories, which has been widely used in the training and evaluations of novel view synthesis and 3D generation or reconstruction methods. Recently, MVImgNet [YXZ’23] presents a large-scale dataset of multi-view images that collects 6.5 million frames from 219,188 videos by shooting videos of real-world objects in human daily life. Other lines of work provide the multi-view dataset in small-scale RGB-D videos [LBRF11, SHG’22, CX’23] compared with these works, large-scale synthetic videos [TME’22], or egocentric videos [ZXA’23]. A large-scale dataset is still a remarkable trend for deep learning methods, especially for generation tasks.

5.3. Learning from Single-view Images

3D generation methods usually rely on multi-view images or 3D ground truth to supervise the reconstruction and generation of 3D representation. Synthesizing high-quality multi-view images or 3D shapes using only collections of single-view images is a challenging problem. Benefiting from the unsupervised training of generative adversarial networks, 3D-aware GANs are introduced that could learn 3D representations in an unsupervised way from natural images. Therefore, several single-view image datasets are proposed and commonly used for these 3D generation methods. Although many large-scale image datasets have been presented for 2D generation, it is hard to directly use them for 3D generation due to the high uncertainty of this problem. Normally, these image datasets only contain a specific category or domain. FFHQ [KL19], a real-world human face dataset consisting of 70,000 high-quality images at 1024\(^2\) resolution, and AFHQ [CUYH’20], an animal face dataset consisting of 15,000 high-quality images at 512\(^2\) resolution, are introduced for 2D image synthesis and used a lot for 3D generation based on 3D-aware GANs. In the domain of the human body, SHHQ [FLJ’22] and DeepFashion [LLQ’16] have been adopted for 3D human generation. In terms of objects, many methods [LSMG20, GMW17, HMR19a, ZZ’18, WZX’16] render synthetic single-view datasets using several major object categories of ShapeNet. While GRAF [SLNG20] renders 150k Chairs from Photoshapes [PRFS18]. Moreover, CelebA [LLWT15] and Cats [ZST08] datasets are also commonly used to train the models like HoloGAN [NPLT19] and pi-GAN [CMK’21a]. Since the single-view images are easy to obtain, these methods could collect their own dataset for the tasks.

6. Applications

In this section, we introduce various 3D generation tasks (Sec. 6.1-6.3) and closely related 3D editing tasks (Sec. 6.4). The generation tasks are divided into three categories, including 3D human generation (Sec. 6.1), 3D face generation (Sec. 6.2), and generic object and scene generation (Sec. 6.3).

6.1. 3D Human Generation

With the emergence of the metaverse and the advancements in virtual 3D social interaction, the field of 3D human digitization and generation has gained significant attention in recent years. Different from general 3D generation methods that focus on category-free rigid objects with sample geometric structures [PBJM23, LXZ’23], most 3D human generation methods aim to tackle the complexities of articulated pose changes and intricate geometric details of clothing. Tab. 5 presents a compilation of notable 3D human body generation methods in recent years, organized according to the input conditions and the output format of the generated 3D human bodies. Some results of these methods are shown in Fig. 8. Specifically, in terms of the input condition, current 3D human body generation methods can be categorized based on the driving factors including latent features randomly sampled from a predefined latent space [MYR’20, CJS’22, HCL’22], a single reference image [APMTM19, CPA’21, XYC’22, HYX’23, ZLZ’23], or text prompts [KKL’23, JWZ’23, KAZ’23, LYX’24]. According to the form of the final output, these methods can be classified into two categories: textureless shape generation [APMTM19,
Although these methods achieve the generation of detailed clothed human shapes, their application prospects are greatly restricted due to the lack of texture-generation capabilities. To generate textured clothed 3D human, lots of attempts have been made in previous work, including methods conditioned on latent codes [GII*21, BKY*22, ZJY*22, NSLH22, JJW*23, YLWD22, XKJ*23, CHB*23, HCL*22, XYB*23, AYS*23], single images [SHN*19, ZYL21D, AZS22, CMA*22, GLZ*23, HYL*23, YLX*23, HHP*23, AST*23, HYX*23, ZLZ*23], and text prompts [HZP*22, JWZ*23, CCH*23, HWZ*23, KAZ*23, ZCY*23, LYX*24, HSZ*23, ZZZ*23a, LST*23]. Most latent-conditioned methods employ adversarial losses to restrict their latent space and generate 3D human bodies within the relevant domain of the training dataset. For example, StylePeople [GII*21] combines StyleGAN [KLA*20] and neural rendering to design a joint generation framework trained in an adversarial fashion on the full-body image datasets. Furthermore, GNARF [BKY*22] and AvatarGen [ZJY*22] employ tri-planes as the 3D representation and replace the neural rendering with volume rendering to enhance the view-consistency of rendered results. To improve editability, Get3DHuman [XKJ*23] divides the human body generation framework into shape and texture branches respectively conditioned on shape and texture latent codes, achieving re-texturing. EVA3D [HCL*22] divides the generated human body into local parts to achieve controllable human poses.

As text-to-image models [RKH*21b, RBL*22b, SCS*22] continue to advance rapidly, the field of text-to-3D has also reached its pinnacle of development. For the text-driven human generation, existing methods inject priors from pre-trained text-to-image models into the 3D human generation framework to achieve text-driven textured human generation, such as AvatarCLIP [HZP*22], AvatarCraft [JWZ*23], DreamHuman [KAZ*23], and TADA [LYX*24]. Indeed, text-driven human generation methods effectively address the challenge of limited 3D training data and significantly enhance the generation capabilities of 3D human assets. Nevertheless, in contrast to the generation of unseen 3D humans, it is also significant to generate a 3D human body from a specified single image in real-life applications. In terms of single-image-conditioned 3D human generation methods, producing generated results with textures and geometries aligned with the input reference image is widely studied. To this end, PiFu [SHN*19], PaMIR [ZYL21D], and PHORHUM [AZS22] propose learning-based 3D generators trained on scanned human datasets to infer human body geometry and texture from input images. However, their performance is constrained by the limitations of the training data. Consequently, they struggle to accurately infer detailed textures and fine geometry from in-the-wild input images, particularly in areas that are not directly visible in the input. To achieve data-free 3D human generation, ELICIT [HYL*23], Human-SGD [AST*23], TeCH [HYX*23], and HumanRef [ZLZ*23] leverage priors of pre-trained CLIP [RKH*21b] or image diffusion models [RBL*22b, SCS*22] to predict the geometry and texture based on the input reference image without the need for 3D datasets, and achieve impressive qualities in generated 3D clothed human.
6.2. 3D Face Generation

One essential characteristic of 3D face generation tasks is to generate high-quality human face images that can be viewed from different viewpoints. Popular tasks can be loosely classified into three major categories, including personalized head avatar creation (e.g., 3D talking head generation), neural implicit 3D morphable models (3DMMs), and generative 3D face models, which are shown in Fig. 9 and Fig. 10.

**Personalized head avatar creation** aims at creating an animatable avatar that can be viewed from different viewpoints of the target person, which has broad applications such as talking head generation. Most of the existing methods take as input a sequence of video frames (i.e., monocular video) [PSB*21, GTZN21, GPL*22, ZAB*22, ZBT23, ZYW*23, BTH*23, GZX*22]. Although convenient, the viewing angles of these avatars are limited in a relatively small range (i.e., near frontal) and their quality is not always satisfactory due to limited data. In contrast, another stream of works [LSSS18, MSS*21, LSS*21, WKC*23, KQG*23] aims at creating a very high-quality digital human that can be viewed from larger angles (e.g., side view). These methods usually require high-quality synchronized multi-view images under even illumination. However, both streams rely heavily on implicit or hybrid neural representations and neural rendering techniques. The quality and animation accuracy of the generated talking head video are usually measured with PSNR, SSIM, and LPIPS metrics.

**Neural implicit 3DMMs.** Traditional 3D morphable face models (3DMMs) assume a predefined template mesh (i.e., fixed topology) for the geometry and have explored various modeling methods including linear models (e.g., PCA-based 3DMMs) and non-linear models (e.g., network-based 3DMMs). A comprehensive survey of these methods has been discussed in [EST*20]. Recently, thanks to the rapid advances in implicit neural representations (INRs), several neural implicit 3DMMs utilizing INRs for face modeling emerge [YTB*21, ZYHC22, GKG*23] since continuous implicit neural representations do not face discretization error and can theoretically modeling infinite details. Indeed, NPHM [GKG*23] can generate more subtle expressions unseen in previous mesh-based 3DMMs. What’s more, neural implicit 3DMMs can potentially model hair better since the complexity of different hairstyles varies drastically, which imposes a great challenge for fixed topology mesh-based traditional 3DMMs.

**Generative 3D face models.** One key difference from 2D generative face models (e.g., StyleGAN [KLA19, KLA*20]) is that 3D face models can synthesize multi-view consistent images (i.e., novel views) of the same target (identity and clothes). Early attempts towards this direction include HoloGAN [NPLT*19] and PlatonicGAN [HMR19b], which are both voxel-based methods and can only generate images in limited resolution. Quickly, methods [SLNG20, NG21, CMK*21b, OELS*22, GLWT22, CLC*22] utilizing neural radiance fields are proposed to increase the image resolution. For example, EG3D [CLC*22] proposes a hybrid triplane representation, which strikes a good trade-off to effectively address the memory and rendering inefficiency faced by previous generative 3D GANs and can produce high-quality images with good multi-view consistency.
Table 6: Applications of general scene generation methods.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Type</th>
<th>Condition</th>
<th>Texture Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVD [ZDW21]</td>
<td>Object-Centered</td>
<td>Latent</td>
<td>×</td>
</tr>
<tr>
<td>NFD [SCP*23]</td>
<td>Object-Centered</td>
<td>Latent</td>
<td>×</td>
</tr>
<tr>
<td>Point-E [NJD*22]</td>
<td>Object-Centered</td>
<td>Text</td>
<td>×</td>
</tr>
<tr>
<td>Diffusion-SDF [LDZL23]</td>
<td>Object-Centered</td>
<td>Text</td>
<td>×</td>
</tr>
<tr>
<td>Deep3Sketch+ [CFZ*23]</td>
<td>Object-Centered</td>
<td>Sketch</td>
<td>✓</td>
</tr>
<tr>
<td>Zero-1-to-3 [LWH*23]</td>
<td>Object-Centered</td>
<td>Single-Image</td>
<td>✓</td>
</tr>
<tr>
<td>Make-It-3D [TW*23]</td>
<td>Object-Centered</td>
<td>Single-Image</td>
<td>✓</td>
</tr>
<tr>
<td>GET3D [GSW*22]</td>
<td>Object-Centered</td>
<td>Latent</td>
<td>✓</td>
</tr>
<tr>
<td>EG3D [CLC*22]</td>
<td>Object-Centered</td>
<td>Latent</td>
<td>✓</td>
</tr>
<tr>
<td>CLIP-Mesh [MKXBP22]</td>
<td>Object-Centered</td>
<td>Text</td>
<td>✓</td>
</tr>
<tr>
<td>ProlificDreamer [WLW*23]</td>
<td>Object-Centered</td>
<td>Text</td>
<td>✓</td>
</tr>
<tr>
<td>DiffDreamer [CCP*23]</td>
<td>Outward-Facing</td>
<td>Single-Image</td>
<td>✓</td>
</tr>
<tr>
<td>Xiang et al. [XHT*23]</td>
<td>Outward-Facing</td>
<td>Latent</td>
<td>✓</td>
</tr>
<tr>
<td>CC3D [BPP*23]</td>
<td>Outward-Facing</td>
<td>Layout</td>
<td>✓</td>
</tr>
<tr>
<td>Text2Room [HCO*23]</td>
<td>Outward-Facing</td>
<td>Text</td>
<td>✓</td>
</tr>
<tr>
<td>Text2NeRF [ZLW*23]</td>
<td>Outward-Facing</td>
<td>Text</td>
<td>✓</td>
</tr>
</tbody>
</table>

stream applications (e.g., editing, talking head generation) are enabled or become less data-hungry, including 3D consistent editing [SWW*23, SWZ*22, SWS*22, LFLSY*23, JCL*22], 3D talking head generation [BFW*23, XSI*23, WLY*23], etc.

6.3. General Scene Generation

Different from 3D human and face generation, which can use existing prior knowledge such as SMPL and 3DMM, general scene generation methods are more based on the similarity of semantics or categories to design a 3D model generation framework. Based on the differences in generation results, as shown in Fig. 11 and Tab. 6, we further subdivided general scene generation into object-centered asset generation and outward-facing scene generation.

6.3.1. Object-Centered Asset Generation

The field of object-centered asset generation has seen significant advancements in recent years, with a focus on both textureless shape generation and textured asset generation. For the textureless shape generation, early works use GAN-based networks to learn a mapping from latent space to 3D object space based on specific categories of 3D data, such as 3D-GAN [WZX*16], HoloGAN [NPLT*19], and PlatonGAN [HMR19b]. However, limited by the generation capabilities of GANs, these methods can only generate rough 3D assets of specific categories. To improve the quality of generated results, SingleShapeGen [WZ22] leverages a pyramid of generators to generate 3D assets in a coarse to fine manner. Given the remarkable achievements of diffusion models in image generation, researchers are directing their attention towards the application of diffusion extensions in the realm of 3D generation. Thus, subsequent methods [LH21, ZDW21, HLHFP22, SCP*23, EMS*23] explore the use of diffusion processes for 3D shape generation from random noise. In addition to these latent-based methods, another important research direction is text-driven 3D asset generation [CCS*19, LWQF22]. For example, 3D-LDM [NKR*22], SDFusion [CLT*23], and Diffusion-SDF [LDZL23] achieve text-to-3D shape generation by designing the diffusion process in 3D feature space. Due to such methods requiring 3D datasets to train the diffusion-based 3D generators, they are limited to the training data in terms of the categories and diversity of generated results. By contrast, CLIP-Forge [SCL*22], CLIP-Sculptor [SFL*23], and Michelangelo [ZLC*23] directly employ the prior of the pre-trained CLIP model to constrain the 3D generation process, effectively improving the generalization of the method and the diversity of generated results. Unlike the above latent-conditioned or text-driven 3D generation methods, to generate 3D assets with expected shapes, there are some works [HMR19a, CFZ*23] that employ image or sketch-conditioned generation.

In comparison to textureless 3D shape generation, textured 3D asset generation not only produces realistic geometric structures but also captures intricate texture details. For example, HoloGAN [NPLT*19], GET3D [GSW*22], and EG3D [CLC*22] employ GAN-based 3D generators conditioned on latent vectors to produce category-specific textured 3D assets. By contrast, text-driven 3D generation methods rely on the prior knowledge of pre-trained large-scale text-image models to enable category-free 3D asset generation. For instance, CLIP-Mesh [MKXBP22], Dream Fields [JMB*22], and PureCLIPNeRF [LC22] employ the prior of CLIP model to constrain the optimization process and achieve text-driven 3D generation. Furthermore, DreamFusion [PJBM23] and SJC [WDL*23] propose a score distillation sampling (SDS) method to achieve 3D constraint which priors extracted from pre-trained 2D diffusion models. Then, some methods further improve the SDS-based 3D generation process in terms of generation quality, multi-face problem, and optimization efficiency, such as Magic3D [LGT*23], Latent-NeRF [MRP23b], Fantasia3D [CCJJ23], DreamBooth3D [RK*23], HiFa [ZZ23], ATT3D [LXZ*23], ProlificDreamer [WLW*23], IT3D [CZY*23], DreamGaussian [TRZ*23], and CAD [WPH*23]. On the other hand, distinct from text-driven 3D generation, single-image-conditioned 3D generation is also a significant research direction [LWH*23, MKLRV23, CGC*23, WLY*23, KDJ*23].

6.3.2. Outward-Facing Scene Generation

Early scene generation methods often require specific scene data for training to obtain category-specific scene categories, such as GAUDI [BGA*22] and the work of Xiang et al. [XHT23], or implement a single scene reconstruction based on the input image, such as PixelSynth [RFJ21] and Worldsheet [HRBP21]. However, these methods are either limited by the quality of the generation or by the extensibility of the scene. With the rise of diffusion models in image inpainting, various methods are beginning to use the scene completion capabilities of diffusion models to implement scene generation tasks [CCP*23, HCO*23, ZLW*23]. Recently, SceneScape [FAKD23], Text2Room [HCO*23], Text2NeRF [ZLW*23], and LucidDreamer [CLN*23] propose progressive inpainting and updating strategies for generating realistic 3D scenes using pre-trained diffusion models. SceneScape and Text2Room utilize explicit polygon meshes as their
Figure 11: Some examples of general scene generation methods. 3D generation results source from Deep3DSketch+ [CFZ*23], NFD [SCP*23], Diffusion-SDF [LDZL23], Make-It-3D [TWZ*23], GET3D [GSW*22], ProlificDreamer [WLW*23], Diffusion-Dreamer [CCP*23], CC3D [BPP*23], Xiang et al. [XYHT23], Text2NeRF [ZLW*23], and LucidDreamer [CLN*23].

3D representation during the generation procedure. However, this choice of representation imposes limitations on the generation of outdoor scenes, resulting in stretched geometry and blurry artifacts in the fusion regions of mesh faces. In contrast, Text2NeRF and LucidDreamer adopt implicit representations, which offer the ability to model fine-grained geometry and textures without specific scene requirements. Consequently, Text2NeRF and LucidDreamer can generate both indoor and outdoor scenes with high fidelity.

6.4. 3D Editing

Based on the region where editing happens, we classify the existing works into global editing and local editing.

6.4.1. Global Editing

Global editing works aim at changing the appearance or geometry of the competing 3D scene globally. Different from local editing, they usually do not intentionally isolate a specific region from a complete and complicated scene or object. Most commonly, they only care if the resultant scene is in a desired new “style” and resembles (maintains some features of) the original scene. Most representative tasks falling into this category include stylization [HTS’21, HHY’22, FJW’22, ZKB’22, WJC’23, HTE’23], and single-object manipulation (e.g. re-texturing [MBOL’22, LZJ’22, MRP’23b, CCJJ23]) as shown in Fig. 12.

Stylization. Early 3D scene stylization methods [HTS’21, HHY’22, FJW’22, ZKB’22] usually require style images to provide style reference. The 3D scene is optimized either in the style feature space using a Gram matrix [GEB16] or nearest neighbor feature matching [ZKB’22] loss or in the image space using the output color of a deep image style transfer network [HB17]. Later methods [WJC’23, HTE’23] can support textual format style definition by utilizing the learned prior knowledge from large-scale language-vision models such as CLIP [RKH*21a] and Stable Diffusion [RBL*22a]. Other than commonly seen artistic style transfer, there also exist some special types of “style” manipulation tasks such as seasonal and illumination manipulations [LLF’23, CZL’22, HTE’23, CYL’22] and climate changes.

Single-Object Manipulation. There are many papers specifically aimed at manipulating a single 3D object. For example, one representative task is texturing or painting a given 3D object (usually in mesh format) [MBOL’22, LZJ’22, MRP’23b, CCJJ23, CSL’23]. Except for diffuse albedo color and vertex displacement [MBOL’22, MZS’23, LYX’24], other common property maps may be involved, including normal map [CCJJ23, LZJ’22], roughness map [CCJJ23, LZJ’22], specular map [CCJJ23, LZJ’22], and metallic map [CCJJ23], etc. A more general setting would be directly manipulating a NeRF-like object [WCH’22, LZJ’22, TLYCS22, YBZ’22]. Notably, the human face/head is one special type of object that has drawn a lot of interest [ATDN23, ZQL’23]. In the meanwhile, many works focus on fine-grained local face manipulation, including expression and appearance manipulation [SWZ’22, SWS’22, LLFSY’23, JCL’22, WDD*22, XSJ’23, MLL’22a, ZLW’22] and face swapping [LMY*23] since human face related understanding tasks (e.g. recognition, parsing, attribute classification) have been extensively studied previously.

6.4.2. Local Editing

Local editing tasks intentionally modify only a specific region, either manually provided ( [MPS’23, LDS’23, CYW’23]) or automatically determined ( [YZX’21, WLC’22, WWL’23, CMS22, JKK’23]), of the complete scene or object. Common
local editing types include appearance manipulation [YBZ*22, ZWL*23], geometry deformation [JKK*23, PYL*22, YSL*22, TLYCS22], object/semantic-level duplication/deletion and moving/removing [YZX*21, WLC*22, KMS22, WWL*23]. For example, NeuMesh [YBZ*22] supports several kinds of texture manipulation including swapping, filling, and painting since they distill a NeRF scene into a mesh-based neural representation. NeRFShop [JKK*23] and CageNeRF [PYL*22] transform/deform the volume bounded by a mesh cage, resulting in moved or deformed/articulated object. SINE [BZY*23] updates both the NeRF geometry and the appearance with geometry prior and semantic (image feature) texture prior as regularizations.

Another line of works (e.g. ObjectNeRF [YZX*21], ObjectSDF [WLC*22], DFF [KMS22]) focus on automatically decomposing the scene into individual objects or semantic parts during reconstruction, which is made possible by utilizing extra 2D image understanding networks (e.g. instance segmentation), and support subsequent object-level manipulations such as re-coloring, removal, displacement, duplication.

Recently, it is possible to create new textures and/or content only according to text description in the existing 3D scenes due to the success of large-scale text-to-image models (e.g. Stable Diffusion [RBL*22a]). For example, instruct-NeRF2NeRF [HTE*23] iteratively updates the reference dataset images modified by a dedicated diffusion model [BHE23] and the NeRF model. DreamEditor [ZWL*23] performs local updates on the region located by text attention guided by score distillation sampling [PJB23]. Focal-Dreamer [LDS*23] creates new geometries (objects) in the specified empty spaces according to the text input. SKED [MPS*23] supports both creating new objects and modifying the existing part located in the region specified by the provided multi-view sketches.

7. Open Challenges

The quality and diversity of 3D generation results have experienced significant progress due to advancements in generative models, 3D representations, and algorithmic paradigms. Considerable attention has been drawn to 3D generation recently as a result of the success achieved by large-scale models in natural language processing and image generation. However, numerous challenges remain before the generated 3D models can meet the high industrial standards required for video games, movies, or immersive digital content in VR/AR. In this section, we will explore some of the open challenges and potential future directions in this field.

Evaluation. Quantifying the quality of generated 3D models objectively is an important and not widely explored problem. Using metrics such as PSNR, SSIM, and F-Score to evaluate rendering and reconstruction results requires ground truth data on the one hand, but on the other hand, it can not comprehensively reflect the quality and diversity of the generated content. In addition, user studies are usually time-consuming, and the study results tend to be influenced by the bias and number of surveyed users. Metrics that capture both the quality and diversity of the results like FID can be applied to 3D data, but may not be always aligned with 3D domain and human preferences. Better metrics to judge the results objectively in terms of generation quality, diversity, and matching degree with the conditions still need further exploration.

Dataset. Unlike language or 2D image data which can be easily captured and collected, 3D assets often require 3D artists or designers to spend a significant amount of time using professional software to create. Moreover, due to the different usage scenarios and creators’ personal styles, these 3D assets may differ greatly in scale, quality, and style, increasing the complexity of 3D data. Specific rules are needed to normalize this diverse 3D data, making it more suitable for generation methods. A large-scale, high-quality 3D dataset is still highly desirable in 3D generation. Meanwhile, exploring how to utilize extensive 2D data for 3D generation could also be a potential solution to address the scarcity of 3D data.

Representation. Representation is an essential part of the 3D generation, as we discuss various representations and the associated methods in Sec. 3. Implicit representation is able to model complex geometric topology efficiently but faces challenges with slow optimization; explicit representation facilitates rapid optimization convergence but struggles to encapsulate complex topology and demands substantial storage resources; Hybrid representation attempts to consider the trade-off between these two, but there are...
still shortcomings. In general, we are motivated to develop a representation that balances optimization efficiency, geometric topology flexibility, and resource usage.

**Controllability.** The purpose of the 3D generation technique is to generate a large amount of user-friendly, high-quality, and diverse 3D content in a cheap and controllable way. However, embedding the generated 3D content into practical applications remains a challenge: most methods \([\text{PJBM23; CLC*’22; YHH*’19b}]\) rely on volume rendering or neural rendering, and fail to generate content suitable for rasterization graphics pipeline. As for methods \([\text{CCJJ23; WLW*’23; TRZ*’23}]\) that generate the content represented by polygons, they do not take into account layout (e.g. the rectangular plane of a table can be represented by two triangles) and high-quality UV unwrapping and the generated textures also face some issues such as baked shadows. These problems make the generated content unfavorable for artist-friendly interaction and editing. Furthermore, the style of generated content is still limited by training datasets. Furthermore, the establishment of comprehensive toolchains is a crucial aspect of the practical implementation of 3D generation. In modern workflows, artists use tools (e.g. LookDev) to harmonize 3D content by examining and contrasting the relighting results of their materials across various lighting conditions. Concurrently, modern Digital Content Creation (DCC) software offers extensive and fine-grained content editing capabilities. It is promising to unify 3D content produced through diverse methods and establish tool chains that encompass abundant editing capabilities.

**Large-scale Model.** Recently, the popularity of large-scale models has gradually affected the field of 3D generation. Researchers are no longer satisfied with using distillation scores that use large-scale image models as the priors to optimize 3D content, but directly no longer satisfied with using distillation scores that use large-scale models. Large-scale Model schemes remain a challenge: most methods \([\text{PJBM23; CLC*’22; YHH*’19b}]\) rely on volume rendering or neural rendering, and fail to generate content suitable for rasterization graphics pipeline. As for methods \([\text{CCJJ23; WLW*’23; TRZ*’23}]\) that generate the content represented by polygons, they do not take into account layout (e.g. the rectangular plane of a table can be represented by two triangles) and high-quality UV unwrapping and the generated textures also face some issues such as baked shadows. These problems make the generated content unfavorable for artist-friendly interaction and editing. Furthermore, the style of generated content is still limited by training datasets. Furthermore, the establishment of comprehensive toolchains is a crucial aspect of the practical implementation of 3D generation. In modern workflows, artists use tools (e.g. LookDev) to harmonize 3D content by examining and contrasting the relighting results of their materials across various lighting conditions. Concurrently, modern Digital Content Creation (DCC) software offers extensive and fine-grained content editing capabilities. It is promising to unify 3D content produced through diverse methods and establish tool chains that encompass abundant editing capabilities.

8. Conclusion

In this work, we present a comprehensive survey on 3D generation, encompassing four main aspects: 3D representations, generation methods, datasets, and various applications. We begin by introducing the 3D representation, which serves as the backbone and determines the characteristics of the generated results. Next, we summarize and categorize a wide range of generation methods, creating an evolutionary tree to visualize their branches and developments. Finally, we provide an overview of related datasets, applications, and open challenges in this field. The realm of 3D generation is currently witnessing explosive growth and development, with new work emerging every week or even daily. We hope this survey offers a systematic summary that could inspire subsequent work for interested readers.

**References**


Advances in 3D Generation: A Survey


LLL∗14] LI B., LU Y., LI C., GODIL A., SCHRECK T., AONO M., CHEN Q., CHOWDHURY N. K., FANG B., FURUYA T., ET AL.:
Shrec’14 track: Large scale comprehensive 3d shape retrieval. In Eurographics Workshop on 3D Object Retrieval (2014), vol. 2, 14


ICCV (2023), 19, 20


[RWC*19] RADFORD A., WU J., CHILD R., LUAN D., AMODEI D., SUTSKEVER I., ET AL.: Language models are unsupervised multitask learners. OpenAI blog 1, 8 (2019). 9, 2, 6, 10


IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021), pp. 7495–7504. 5


[XKJ‘23] Xiong Z., Kang D., Jin D., Chen W., Bao L., Cui S., Han X.: Get3dhuman: Lifting stylegan-human into a 3d generative model using pixel-aligned reconstruction priors. In Proceedings of
the IEEE/CVF International Conference on Computer Vision (2023), pp. 9287–9297. 15, 16


