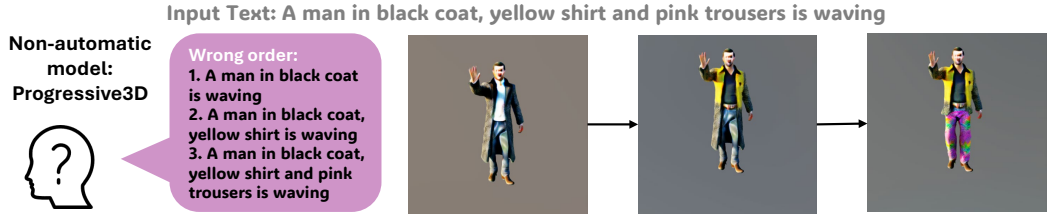
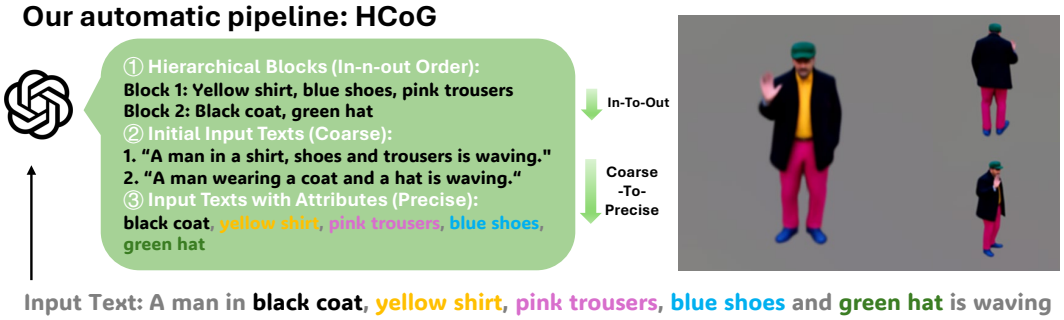
Apply Hierarchical-Chain-of-Generation to Complex Attributes Text-to-3D Generation Supplementary Material

Yiming Qin Zhu Xu Yang Liu*
Wangxuan Institute of Computer Technology, Peking University
kevinqym@stu.pku.edu.cn xuzhu@stu.pku.edu.cn yangliu@pku.edu.cn

(a) Visualization of cross-attention map. For longer text prompts, 2D Stable Diffusion (SD) [27] fails to accurately associate the word "black" with the correct spatial location in the generated image. This limitation poses a challenge for methods [16, 28] lifting 2D to 3D using SD effectively.



(b) Some work such as Progressive3D [5] that targets 3D generation with complex attributes heavily relies on user-defined bounding boxes and generation order, and imperfect given order results in low-quality results with wrong attributes.



(c) Our method (HCoG) leverages LLM to generate hierarchical chain of generation, realizing automatic generation of 3D assets with better complex attributes binding capability.

Figure 1. **The problem of existing work and the example of our method.**

Abstract

Recent text-to-3D generation models have demonstrated remarkable abilities in producing high-quality 3D assets. Despite their great advancements, current models struggle to generate satisfying 3D objects with complex attributes. The difficulty for such complex attributes 3D generation arises from two aspects: (1) existing text-to-3D approaches typically lift text-to-image models to extract semantics via text encoders, while the text encoder exhibits limited comprehension ability for long descriptions, leading to deviated cross-attention focus, subsequently wrong attribute binding in generated results. (2) Objects with complex attributes often exhibit occlusion relationships between different parts, which demands a reasonable generation order as well as explicit disentanglement of different parts to enable structural coherent and attribute following results. Though some

* Corresponding author.

*works introduce manual efforts to alleviate the above issues, their quality is unstable and highly reliant on manual information. To tackle above problems, we propose a automated method **H**ierarchical-**C**hain-**o**f-**G**eneration (**HCoG**). It leverages a large language model to analyze the long description, decomposes it into several blocks representing different object parts, and organizes an optimal generation order from in to out according to the occlusion relationship between parts, turning the whole generation process into a hierarchical chain. For optimization within each block, we first generate the necessary components coarsely, then bind their attributes precisely by target region localization and corresponding 3D Gaussian kernel optimization. For optimization between blocks, we introduce Gaussian Extension and Label Elimination to seamlessly generate new parts by extending new Gaussian kernels, re-assigning semantic labels, and eliminating unnecessary kernels, ensuring that only relevant parts are added without disrupting previously optimized parts. Experiments validate HCoG's effectiveness in handling complex attributes 3D assets and witnesses high-quality results. The code is available at https://github.com/Wakals/GASCOL.*

1. Introduction

In the field of 3D vision, the development of user-friendly generation of 3D assets with complex attributes has received gaining attention. It allows users to achieve the expected personalized 3D asset generation with few manual efforts. For example, Shap-e [10] uses the transformer to achieve direct 3D generation. Later works such as DreamFusion [25] and SJC [31] propose to lift prior information in text-to-2D model to 3D, which greatly improves the generalization and detail of the generated results. Since the prior knowledge of 2D diffusion is pervasive, the 2D-based method has stronger generalization and is not limited by the small size of 3D data. Therefore, our work focuses on the method of generating 3D with the help of 2D text-to-image diffusion.

Though significant progress for text-to-3D models, when encountering objects with complex attributes, the results quality of previous approaches like LucidDreamer [16] and MVDream [28] still lag behind with attributes deviation, like shown in Fig. 1a, the cross-attention maps fail to attend to the correct object regions. We claim two primary reasons for such difficulty in complex attributes text-to-3D generation: Firstly, the widely used CLIP [26] text encoder struggles to accurately encode long descriptions, as noted in prior research [37], and may overlook crucial information, leading the cross-attention maps fail to correctly align attribute descriptions with their corresponding image regions. For instance, as shown in Fig. 1a, the attention of **black** principally clusters on the head and the shirt, not the coat. Such deviated attentions finally result in wrong attribute binding in generated 3D objects. Secondly, objects with complex attributes naturally exhibit occlusion relationships between different object parts, which requires a reasonable generation order as well as explicit disentangled optimization for these parts to enable structural-coherent and attribute-following results. Some more recent works like Progressive3D [5] propose to handle such complex attribute objects by introducing user-defined generation order as well as bounding boxes as guidance. But such manual efforts introduction hinder the generation automation, and the quality of the result heavily relies on user-provided generation order, where an incorrect order of generation will reveal serious consequences as shown in Fig. 1b. Such failure originates from prioritizing the generation of the external part with less occlusion, during the subsequent generation of the internal part with more occlusion, the surface of the previously generated external part was affected. Thus such manual information-guided generation is also not optimal for complex attributes text-to-3D generation task.

To tackle the above problems and enable high-quality complex attributes text-to-3D generation, we introduce Hierarchical-Chain-of-Generation (HCoG), which generates complext attributes 3D assets in the representation of 3D Gaussian Splatting [11] (3DGS). Three key designs underpin our HCoG framework: (1) **Hierarchical Blocks:** we employ a LLM to analyze the text description and decompose the whole object into different parts with shorter descriptions as hierarchical blocks for separate generation. It ensures the quality is not limited by the comprehension ability of text encoder. Further, we propose a *In-n-out order of generation* strategy, which decides the generation order by parts' occlusion relationships, and prioritize to generation inner blocks, which not only can fully expose the inner parts that are occluded for better optimization, but also facilitates the structural integrity of outer parts, yielding more structural coherent results. We also introduce a *generate coarsely-to-precisely* paradigm within each block, which first utilizes coarse-grained attribute-agnostic text for necessary component generation, then applies fine-grained attribute-aware text for detailed attributes editing, improving generation efficiency. (2) **Part-optimization:** within each block, after necessary components are generated, we target the fine-grained attribute editing for these components. We first adopt *part segmentation* to precisely locate the target region, choosing the corresponding 3D Gaussian kernels for subsequent *fine-grained optimization* to optimize and bind the attributes, during which we introduce MVDream[28] and ControlNet[38] to facilitate shape control and multi-view consistency for generation. (3) **Gaussian Extension and Label Elimination**: to generate new parts based on previous optimized parts, we first adopt *gaussian extension* to densify the Gaussian kernels to form new parts. To further exclude the affection for previ-

ous optimized parts like changing their appearance during such densification process, we introduce *label elimination*, which re-assigns semantic labels for densified new Gaussian kernels, removing unnecessary kernels and only keep kernels that belong to the new part. Notably, our **Gaussian Extension and Label Elimination** avoid manual input information like user-provided bounding boxes, enabling totally automated generation of HCoG. Besides, HCoG can serve as a plug-and-play generation paradigm for diverse text-to-3D models, showing high scalability. Our main contributions are summarized as follows:

- We propose Hierarchical-Chain-of-Generation (HCoG), a framework that automates the generation of 3D assets with complex attributes via decomposing the object into hierarchical blocks ordered by occlusion relations for sequential generation.
- We propose a coarse-to-fine optimization approach to achieve faithful attribute binding within each hierarchical block. Further, we introduce a Gaussian Extension and Label Elimination strategy between blocks, which eliminates the requirements for manual input guidance for generation and ensures the new-part generation will not affect the optimized parts.
- Experiments show that HCoG can automatically generate high-quality 3D assets with complex attributes, especially with strong occlusion relationships, outperforming previous automatic text-to-3D methods. By applying HCoG on different text-to-3D models, we verify its scalability.

2. Related Work

2.1. Text-to-3D Generation

The generation of 3D objects has garnered significant attention from researchers, with an increasing number of studies[10, 14, 23, 34] focusing on this area. Though great progress, the low quality and scarcity of 3D available data remains a significant challenge to these 3D generation methods. To tackle it, DreamFusion [25] introduced Score Distillation Sampling (SDS) loss, which distills 2D prior knowledge from a pre-trained diffusion model into the 3D domain, optimizing a 3D representation for each input text. Since then, an increasing number of works [2, 4, 7, 15, 17, 19, 29, 31, 32, 36] have been focusing on utilizing some methods to facilitate the process and generate high-fidelity objects. Such as LucidDreamer [16], ProlificDreamer [33] and SDS-Bridge [21] modify SDS loss function to generate higher-quality objects. Among these methods, Progressive3D [5] have made a contribution on the generation of 3D assets with complex attributes. However, it needs plenty of manual work and will fail if users make mistakes about the order of generation and the location of bounding boxes when there is obvious occlusion in target generated asset.

2.2. 3D Editing

When we need more customized 3D assets and 3D data or want to optimize some 3D data, 3D Editing is an essential tool. However, there are certain challenges in editing 3D data precisely. EditNeRF [18] is an early proposal for editing 3D data, which uses coarse 2D user scribbles to edit the neural radiance field. After this work, a large plenty of effort has been put into editing the neural radiance field. SINE [1], TextDeformer [6], CLIP-NeRF [30], ED-NeRF [24] propose one-stage method to edit neural radiance field based on the given text or the reference image. On editing 3D Gaussian Splatting, the pioneering work GaussianEditor [3], Gaussctrl [35] and [20] edit 3D Gaussian Splatting using text prompt or referred images. All these works directly edit 3D scenes based on simple input texts or images. Different from these works, our work aims to decompose complex long input text and leverage the insight of editing to optimize test-to-3D assets with careful consideration of optimization order , which ensures the accurate generation of objects with complex attributes.

3. Preliminaty Knowledge

3.1. 3D Gaussian Splatting

Gaussian Splatting utilizes a set of 3D Gaussian kernels to fit the 3D scene or object, serving as one powerful 3D representation with high-quality. Formally, Gaussian kernels can be parameterized as θ. For each $\theta_i = \{\mathbf{x}_i, \mathbf{s}_i, \mathbf{q}_i, \mathbf{c}_i, \alpha_i\}$, where $\mathbf{x}_i \in \mathbb{R}^3$ represents the coordinate of the center of the i-th Gaussian in Cartesian coordinate system, $\mathbf{s}_i \in \mathbb{R}^3$ represents the scaling size, $\mathbf{q}_i \in \mathbb{R}^4$ is the rotation of the i-th Gaussian which is represented as a quaternion, $\mathbf{c}_i \in \mathbb{R}^3$ contains the RGB of this Gaussian kernel and the $\alpha_i \in \mathbb{R}$ is the opacity value. The whole space is served as tile list to be projected onto the screen plane by a sample camera, and the color $\mathbf{C}(\mathbf{p})$ of each point $\mathbf{p}$ on the projection screen is calculated with the formula:$\mathbf{C}(\mathbf{p}) = \sum_{i\in\mathcal{N}} \mathbf{c}_i \alpha'_i \prod_{j=1}^{i-1}(1-\alpha'_j)$, where $\alpha'_i = \alpha_i e^{-\frac{1}{2}(\mathbf{p}-\mathbf{x}_i)^T \Sigma_i^{-1}(\mathbf{p}-\mathbf{x}_i)}$, Σ_i is the covariance of the i-th Gaussian which can be arrived by $\mathbf{s}_i$ and $\mathbf{q}_i$, and $\mathcal{N}$ denotes the number of Gaussians in this tile. Since 3D Gaussian Splatting is a display expression, we can use the characteristics of the display expression to map each kernel to the corresponding part of the 3D asset, making it easier to optimize a certain part separately.

3.2. Score Distillation Sampling (SDS)

Score Distillation Sampling proposed in DreamFusion [25] aims to distill the prior knowledge in 2D diffusion models for 3D generation. A key advantage of this technology is its independence from 3D data. In the absence of 3D data, SDS technology demonstrates enhanced generalization capabilities and can produce a wider range of diverse results, making it especially beneficial for users seeking to gener-

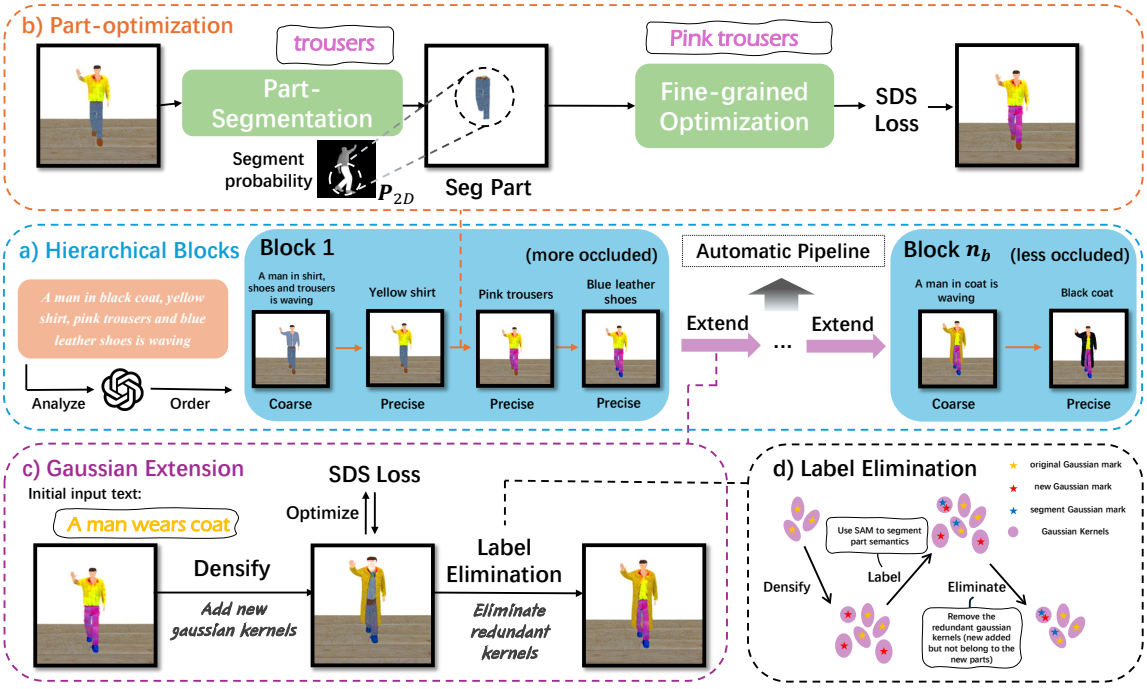


Figure 2. **Overview of Hierarchical-Chain-of-Generation.** a) In the Hierarchical Blocks stage, LLM analyzes the input text and based on the order from more occlusion to less occlusion, creating the order of generation. b) Part-optimization is applied to the parts in blocks, using Lang-SAM [22] to segment specific parts and utilizing MVDream [28] and ControlNet [38] in fine-grained optimization stage to enable corresponding attributes binding for each part with shape and multi-view consistency. c) Gaussian Extension is applied between blocks, extending new parts for the next block. d) Label Elimination aims to generate new parts by extending new Gaussian kernels (red-star-marked), re-assigning semantic labels (blue-star-marked), and eliminating unnecessary kernels finally, ensuring that only relevant parts are generated without disrupting previously optimized parts.

ate personalized 3D assets with complex attributes. In this paradigm, the parameter of the 3D scene θ is denoted as differentiable image parameterization. Then, after rendering image $\mathbf{I}_{RGB}$ according to a given camera pose $\mathbf{p}_c$, a Gaussian noise $\epsilon(0, I)$ is added onto the image and passed into the diffusion model ϕ. With the predicted noise ϵ_ϕ at timestep t, SDS loss optimize the 3D scene parameter θ by calculating the difference between added noise $\epsilon(0, I)$ and the predicted noise ϵ_ϕ, which is formulated as,

$$\nabla_\theta \mathcal{L}_{\text{SDS}} = \mathbb{E}_{\epsilon,t}\left[w(t)\left(\epsilon_\phi(\mathbf{I}_{RGB}; y(\mathbf{p}_c), t) - \epsilon\right)\frac{\partial \mathbf{I}_{RGB}}{\partial \theta}\right] \quad (1)$$

where $y(\mathbf{p}_{camera})$ is the text prompt related to the camera pose $\mathbf{p}_c$ and $w(t)$ is a weight function. To speed up the process of backpropagation, SDS loss simply skips the grad of U-net.

4. Method

The framework of Hierarchical-Chain-of-Generation is shown in Fig. 2, which comprises three main designs. We first adopt **Hierarchical Blocks** (detailed in Sec. 4.1) to analyze the text description and decompose the whole generation into sequential blocks. To yield more structural coherent results, in-n-out order strategy are further introduced to generate these blocks from inner ones to outer ones according to the occlusion relationships between them. Then within each block, we propose **Part Optimization** (detailed in Sec. 4.2) to first coarsely generate necessary components, then precisely edit and bind corresponding attributes through target region localization and fine-grained 3D gaussian kernel optimization, yielding faithfully and accurate attribute binding for each object part. Between blocks, **Gaussian Extension and Label Elimination** (detailed in Sec. 4.3) is introduced to generate new parts, which firstly extend new Gaussian kernels through kernel densification, then re-assign semantic labels for newly generated kernels, and eliminate redundant ones according to labels, which only eliminate the manual efforts to make whole generation automatic, but also avoid negative affection like appearance changing in previous optimized parts.

4.1. Hierarchical Blocks

As shown in Fig. 2, when faced with complex 3D objects with multiple constituent parts and distinct attributes, it is highly probable that certain parts may occlude others, resulting in only partial visibility of some elements from any given viewpoint, thereby confusing 2D diffusion model and complicating the generation of 3D assets. To address the issue of attribute binding and ensure a fully automated pipeline, we propose Hierarchical Blocks, which consist of two key design principles: (1) **In-n-out order of generation:** We first extract all object parts from the long compli-

cated input text and organize them into hierarchical layers according to their occlusion relationships. The most occluded parts are placed in the initial layer and generated first, while the least occluded parts are assigned to the final layer and generated last. Parts that do not occlude one another are grouped within the same layer, allowing for parallel generation. (2) **Generate coarsely-to-precisely:** Within each hierarchical block, all constituent parts are first generated by the initial input text, and then are refined as attribute binding by the fine-grained input texts. Specifically, the initial input text includes only the parts in this block, such as **"a man in shirt, shoes and trousers is waving"** shown in session a) of Fig. 2, is coarsely initial input text, omitting detailed attributes and generating parts in this block. Progressively, **"yellow shirt"** is the precise optimization input text, enhancing the attribute "yellow" bound to "shirt".

In-n-out order of generation. Adopting our generation order ensures more structurally coherent results. Specifically, heavily occluded parts are often entirely or partially invisible in the final rendered image, making post-generation optimization nearly impossible. Therefore, it is crucial that these occluded parts are generated and optimized before the outer parts are introduced. By ensuring that the outer layers remain ungenerated during this process, the system can fully expose the occluded components, allowing them to be accurately reconstructed and optimized. Moreover, the structural integrity of the outer parts depends largely on the shape of the inner parts. Thus, a hierarchical generation strategy, which prioritizes inner (more occluded) parts before outer (less occluded) ones, ensures better global consistency in the final 3D asset. For parts that do not occlude each other, their generation order remains flexible and they can be grouped within the same hierarchical block to enable parallel processing.

Generate coarsely-to-precisely. The generation of all parts in the same block is based on generating all new parts coarsely and editing precisely. In each block, the initial input text is responsible for generating all parts associated with that block simultaneously. Since 3D generation is computationally intensive, producing multiple components in a single forward pass significantly improves efficiency compared to sequential generation. To optimize the generative model's performance, the initial input text excludes attribute-level details, as minimizing textual complexity enhances the model's ability to learn high-level structural information. Once the coarse representation of all parts is established, subsequent fine-grained refinements are performed, where each part's attributes are explicitly defined and serially incorporated into the generation process. An illustrative example is presented in Fig. 1c. The hierarchical structure specifies the grouping of parts into different blocks. The initial input text for each block captures a high-level structural outline of all the parts within that block, such as "A man in a shirt, shoes and trousers is waving" contains all "shirt", "shoes" and "trousers" in this block. Meanwhile, it also provides precise descriptions of each part along with its attributes, such as "yellow shirt", to bind attribute to the corresponding part.

To make this process totally automated, we use a large language model to help us create **Hierarchical Blocks**. Since many objects or parts in daily life have occlusion relationships, the large language model (LLM) has prior knowledge of occlusion relationships, assisting us to analyze the occlusion relationships of various parts in long complex input texts, and further providing the order of generation based on occlusion relationships. In addition, the LLM is capable of extracting various parts and corresponding attributes and providing simplified initial input texts, which is shown in Fig. 1c.

4.2. Part-optimization

Within each hierarchical block, multiple parts coexist without occluding one another. Once the coarse structures are generated, each part undergoes precise optimization, which first requires accurate localization and segmentation. To achieve this, we employ SAM [12] to segment 2D rendered images and lift the 2D to 3D. Subsequently, the fine-grained optimization is based on the framework of MVDream [28] which ensures multi-view consistency, adopting SDS loss to optimize 3D assets which are formulated as Eq. 1.

Part Segmentation is the first stage, aiming to segment certain part that needs optimization and using lang-SAM [22] to supervise the segmentation of the target part. Specifically, we bind a new property on each Gaussian kernel, labeled p_{seg}, randomly initialized, which acts as a binary label, indicating the kernel's possibility of belonging to the target part. The goal of this stage is to optimize p_{seg} of all kernels so that it converges to a point where the kernels belonging to the certain part have higher p_{seg}. For each time the 3D asset is rendered in a sampled camera pose, the binary label p_{seg} is rendered as a 2D tensor $\mathbf{P}_{2D}$, which means the current binary label renders as 2D tensor with segmentation possibility and needed optimization. In the meantime, an RGB image $\mathbf{I}_{RGB}$ is rendered based on the same camera pose. Then, we apply lang-SAM[22] to the rendered image to obtain a segmentation ground-truth $\mathbf{P}_{\mathbf{I}_{RGB}}$. The segment loss is formulated as:

$$\mathcal{L}_{segment} = \text{CrossEntropy}(\mathbf{P}_{2D}, \mathbf{P}_{\mathbf{I}_{RGB}}) \tag{2}$$

Through iterations of training, we yield the converaged binary classification labels p_{seg} to group the Gaussian kernels belonging to the target part. As shown in session b) of Fig. 2, after multi-iteration, the Segment probability converges on the trousers and this part is segmented out.

Fine-grained Optimization is the second step to optimize the target part, aiming to bind correct attributes to new corresponding parts. For each part, we only allow the Gaus-

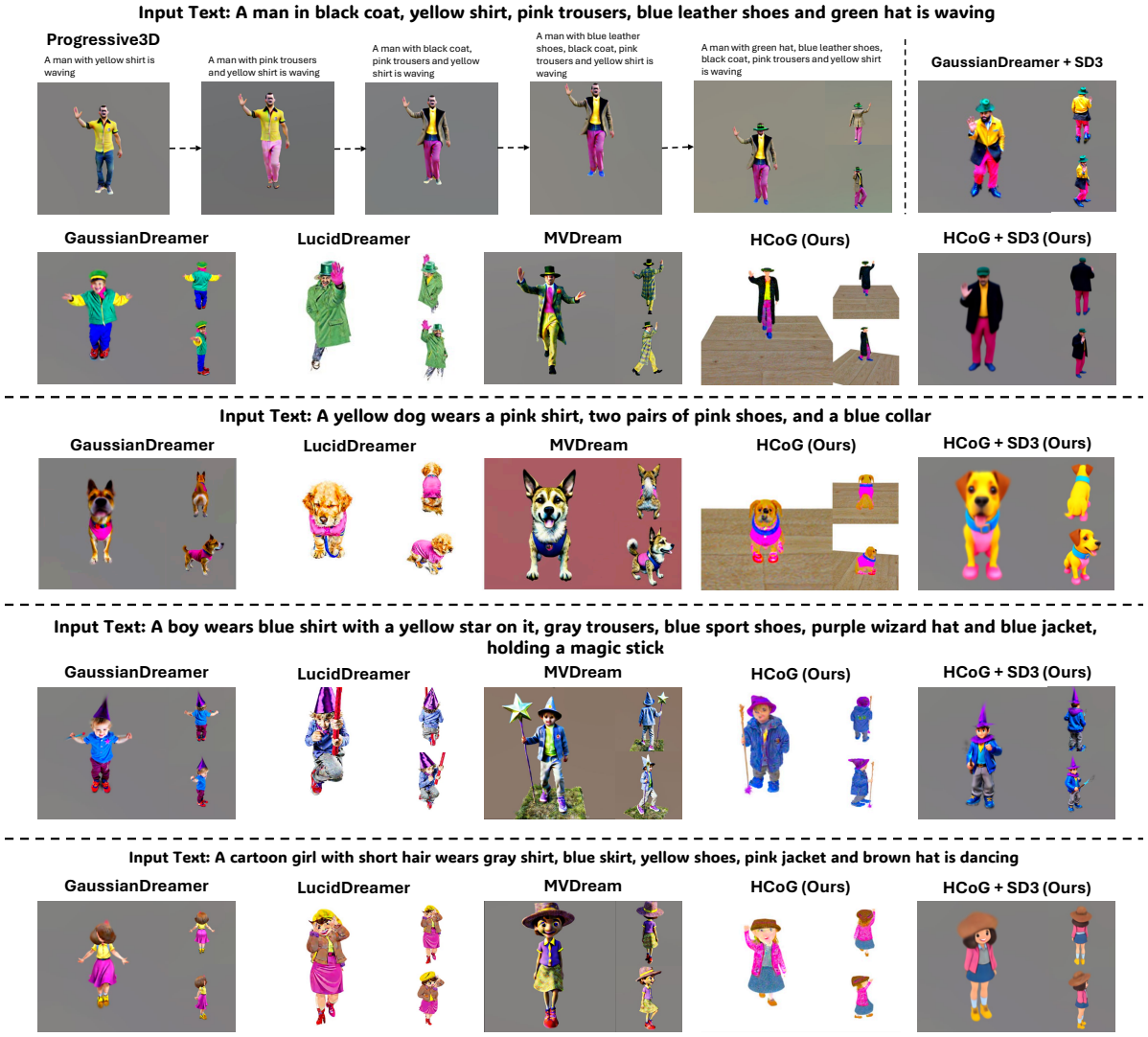


Figure 3. **Visual comparison with other methods.** We compare our method with other well performed text-to-3D methods [16, 28, 36], Progressive3D [5] which heavily relies on user-defined generation order and bounding boxes, and Stable Diffusion v3 [27] which is a more powerful backend.

sian kernels belonging to the target part to receive gradients while fixing the other Gaussian kernels. During the optimization process, since the quality of generated assets may not be optimal at the early stage and some parts show differences in sizes and shapes from prior knowledge of diffusion models, we further introduce to combine ControlNet [38] to provide the shape prior to the generated assets to diffusion, which corrects the error caused by the gap between the generated Gaussian Splatting and the real Gaussian Splatting data. Therefore, the segmented part is fed in to ControlNet [38] for better shape consistency, applied SDS loss:

$$\mathcal{L}_{optim} = \mathcal{L}_{\text{SDS}}^{\text{ControlNet}} + \mathcal{L}_{\text{SDS}}^{\text{MVDream}} \tag{3}$$

where $\mathcal{L}_{\text{SDS}}^{\text{ControlNet}}$ is the $\mathcal{L}_{\text{SDS}}$ from ControlNet [38] and $\mathcal{L}_{\text{SDS}}^{\text{MVDream}}$ is from MVDream [28]. The formula of $\mathcal{L}_{\text{SDS}}$ is referred as Eq. 1.

4.3. Gaussian Extension and Label Elimination

In order to reduce manual efforts like precise user-defined bounding boxes and not change the appearance of previous optimized parts, we propose the Gaussian Extension operation to generate new parts without changing the original Gaussian kernels and the Label Elimination to ensure new parts not change the appearance of previous parts which have been optimized.

Gaussian Extension. The goal of the Gaussian Extension operation is to generate the parts of the next block while preserving the previous parts, and it does not depend on the bounding boxes defined by the user. As shown in Fig 2 Part c), we densify the original 3DGS and get some new Gaussian kernels. For each new Gaussian kernel, it is generated from a Gaussian kernel in the original 3DGS. The properties of the new Gaussian kernel are copied from

the original Gaussian Kernel, except for the position. The position of new kernel $\mathbf{x}_{new}$ will be sampled on the distribution of the original Gaussian kernel $N(\mathbf{x}_{origin}, \mathbf{s}_{origin})$ and a small random perturbation will be added, which can be formulated as:

$$\mathbf{x}_{new} = \mathbf{x}_{sample} + \mathbf{x}_{perturb} \tag{4}$$

where $\mathbf{x}_{origin}$ and $\mathbf{s}_{origin}$ denote the center and the scale of the original gaussian kernel, $\mathbf{x}_{sample} \sim N(\mathbf{x}_{origin}, \mathbf{s}_{origin})$ denotes sampling a point on the distribution of original Gaussian kernel and $\mathbf{x}_{perturb} \sim N(0, \epsilon)$ means random perturbation and ϵ is a tiny noise covariance. We fix all the original 3DGS and only allow the new 3DGS to accept gradients, thus to ensure preserve the original parts. Then, we use the simplified next-block text provided by LLM to optimize all 3DGS via SDS loss, so that all the parts of the next block can be optimised.

Label Elimination. Although the Gaussian Extension method does not require the user to define the precise position of the bounding boxes, it leads to attribute deviation within previous parts, which is shown in Sec. 5.4. The reason is, during Gaussian Extension stage, the extended kernels may be attached the the surface of the previous optimized parts. Therefore, to preserve the appearance of these optimized parts, we propose Label Elimination, a concise and effective way to remove the influence of the parts in the previous block and only keep the new parts in the next block.

Specifically, as shown in Fig. 2, purple circles representing kernels and colorful stars representing marks, before the Gaussian Extension step, all the original gaussian kernels are orange-star-marked. After Gaussian Extension method, the densified Gaussian kernels are red-star-marked. After the optimization, we apply SAM [12] to segment the parts that are novel in this new block. All Gaussian kernels that are segmented as new parts are blue-star-marked. After this operation, the blue-star-marked kernels contain the parts we want to add, while the red-star-marked kernels are actually not belong to the parts we want to add, which means they are redundant. Subsequently, we only need to eliminate the red-star-marked kernels to get a harmonious 3D asset with new parts without negatively affecting the original optimized parts.

Besides, because different Gaussian kernels are required as the size varies for each part, Label Elimination can help remove redundant Gaussian kernels so that during the process of Gaussian Extension, we only need to add a fixed number of Gaussians.

5. Experiments

5.1. Implementation Details

Our hierarchical chain-of-generation (HCoG) framework is designed to be compatible with various backbones. In this paper, we implement HCoG on both GALA3D [39] based on Stable Diffusion v2.1 and GaussianDreamer [36] equipped with the advanced text-to-image model Stable Diffusion v3. We use GPT-4o as the large language model. During the process of **Part Segmbentation**, a threshold is needed to identify which part a Gaussian kernel belongs to and is set to 0.9. p_{seg} is trained for 200 iterations where the learning rate is set to 0.05. The camera's sampling radius is set to the range of the scene in a spherical coordinate system, while vertical angles are sampled uniformly from $-45°$ to $45°$ and horizontal angles are sampled uniformly from $360°$. In the Process of **Extend**, the random perturbation ϵ is set to 0.01.

	GSD [36]	MVD [28]	Pro3D* [5]	Ours	Ours+SD3
BLIP-VQA	0.4919	0.5519	0.6553	<u>0.7295</u>	**0.8055**
CLIP-Score	30.709	31.132	30.451	<u>31.998</u>	**33.189**

Table 1. **Quantitative Comparison.** Where GSD means GaussianDreamer, MVD denotes MVDream, and Pro3D* represents Progressive3D. Our method outperforms other methods.

5.2. Qualitative Comparisons

The comparison of visualized results with other methods is shown in Fig. 3. We choose some well-performed text-to-3D methods [5, 16, 28, 36] to compare with our method (Ours). Besides, we upgrade the 2D diffusion model used as SDS guidance to more advanced Stable Diffusion v3, and compare GaussianDreamer [36] with our method (Ours+SD3), both equipped with Stable Diffusion v3. For testing, we selected challenging examples featuring complex parts and attributes, including those with distinct occlusion relationships, to evaluate the model's performance in complex attribute text-to-3D generation.

Compare with Progressive3D [5]. As shown in the first row of Fig. 3, Progressive3D [5] performs generally satisfactorily and shows competitive results when the input text is complex. However, it needs plenty of manual effort. Users is requested to define the generation order and bounding boxes carefully, which makes it less practical. Besides, if the order is not proper, the results may be a bad crash. As shown in Fig. 1b, generating black coat first and yellow shirt later will dye the coat yellow. In comparison, our effort has the ability to automatically decide the optimization order and generate 3D assets without any manual effort.

Compare with SOTA text-to-3D methods. As shown in Fig 3, we perform visual comparison of our method with GaussianDreamer [36], MVDream [28] and LuciDreamer [16], which are all equipped on the same version of Stable Diffusion as ours. In these cases, admittedly, these methods are able to generate delicate and high-quality results, while they are unable to tackle the task of 3D generation with such complex attributes. They usually bind the parts with wrong attributes in the input text, such as binding the star on the magic stick instead of the shirt, while

	Inverse order	Random order	Ours
HCoG	0.5023	0.5961	**0.7295**
+SD3	0.6363	0.7033	**0.8055**

Table 2. **Ablation of Generation order on BLIP-VQA.**

Figure 4. **Ablation study of ControlNet. Input text: blue sports shoes.** Without shape control, the diffusion model will give wrong guidance and the result will be bad.

Figure 5. **Ablation study of Label Elimination. Previous optimized input text: A man in yellow shirt, pink trousers and blue leather shoes is waving. Next input text: A man in coat is waving.** Without Label Elimination, when generating new part **coat**, the optimized parts like **yellow shirt**, **pink trousers** and **blue leather shoes** are changed.

sometimes missing some parts, such as the two pairs of pink shoes in that dog. Meanwhile, our method have the ability to tackle these situations.

Compare with GaussianDreamer [36] with Stable Diffusion v3. When upgrading the 2D diffusion model to the more advanced, recent Stable Diffusion v3 (as shown in the right-most column), we observe that GaussianDreamer [36] demonstrates some improvement but remains inadequate in attribute binding. It bind "yellow" to "coat" by mistake. In contrast, when equipping Stable Diffusion v3 to HCoG, our method enables more fine-grained details and higher-quality outputs while maintaining the correctness of attribute binding. Note that the order of operations remains critical when upgrading 2D diffusion model to Stable Diffusion v3. Additional ablation studies can be found in Sec. 5.4.

5.3. Quantitative Comparison

In order to quantitatively compare the results, we adopted the BLIP-VQA [13] method proposed in T2I-CompBench [9] and CLIP [26] similarity score to evaluate the quality of the generated results. However, according to the T2I-CompBench [9], the BLIP-VQA scores of each part are multiplied to get the total score. However, due to our set-up of "3D assets with complex attributes" being challenging, we empirically observed that the original BLIP-VQA score of most methods is zero. Therefore, in order to enable a more intuitive comparison, we changed the total score to the average of the scores of each part, and finally obtained the results shown in Tab. 1. As can be seen, our method is better than many previous text-to-3D methods [28, 36], and also outperforms Progressive3D [5] which targets for complex attributes binding by 7.42% on BLIP-VQA and by 1.547 on CLIP-score, achieving the best score. The other advantage is that our method needs no manual effort. Besides, when we take Stable Diffusion v3 as our backend, the score achieves 0.8055 on BLIP-vQA and 33.189 on CLIP-score, which implies stronger 2D diffusion model provides more reliable guidance.

5.4. Ablation Experiments

We conduct an ablation study to evaluate the effectiveness of the order of generation, ControlNet [38] and Label Elimination respectively.

Order of generation. We conduct an ablation experiment on the order of generation and the results is shown in Tab. 2, which reveals that generating in order from the severely obscured to the lightest obscured ensures each part can be well optimized. Even equipped with more advanced SD3, the order of generation is still crucial in generating high quality results.

ControlNet. We conduct experiments with and without ControlNet [38] to verify its effectiveness. As shown in Fig. 4, ControlNet provides shape and size information to the diffusion model, ensuring stable optimization. Empirically, ControlNet [38] is essential for Part-optimization, as it bridges the gap between real and generated data for Gaussian Splatting, preventing issues with distorted shapes and sizes that confuse the diffusion model.

Label Elimination. We conduct experiments with and without Label Elimination to verify its effectiveness. As shown in Fig. 5, which indicates that Label Elimination is the key design to generate new parts without interfering with the previous optimized parts by removing redundant kernels attached to the surface of previous optimized parts.

6. Conclusion

We present a method Hierarchical-Chain-of-Generation (HCoG) that targets complex attributes text-to-3D generation task. It utilizes a LLM to analyze the input text description, decomposes the object into hierarchical blocks with different object parts to generate them sequentially with the order decided by their occlusion relationships. Within each block, a coarse-to-fine optimization process is conducted to faithfully bind attributes for each part. Between blocks, gaussian kernels extension and label elimination are proposed to smoothly generate new parts without disrupting previously optimized ones.The entire pipeline is fully automated, minimizing manual effort and enhancing user-friendliness. Experiments demonstrate the effectiveness and scalability of our method.

Acknowledgment

This work was supported by the grants from the Beijing Natural Science Foundation 4252040 and National Natural Science Foundation of China 62372014.

References

[1] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Sine: Semantic-driven image-based nerf editing with prior-guided editing field. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20919–20929, 2023. 3

[2] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *ICCV*, 2023. 3

[3] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21476–21485, 2023. 3

[4] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585*, 2023. 3

[5] Xinhua Cheng, Tianyu Yang, Jianan Wang, Yu Li, Lei Zhang, Jian Zhang, and Li Yuan. Progressive3d: Progressively local editing for text-to-3d content creation with complex semantic prompts. *ArXiv*, abs/2310.11784, 2023. 1, 2, 3, 6, 7, 8

[6] William Gao, Noam Aigerman, Thibault Groueix, Vladimir G. Kim, and Rana Hanocka. Textdeformer: Geometry manipulation using text guidance. *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 3

[7] Pengsheng Guo, Hans Hao, Adam Caccavale, Zhongzheng Ren, Edward Zhang, Qi Shan, Aditya Sankar, Alexander G. Schwing, Alex Colburn, and Fangchang Ma. Stabledreamer: Taming noisy score distillation sampling for text-to-3d. *ArXiv*, abs/2312.02189, 2023. 3

[8] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *ArXiv*, abs/2104.08718, 2021. 1

[9] Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *ArXiv*, abs/2307.06350, 2023. 8, 1

[10] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *ArXiv*, abs/2305.02463, 2023. 2, 3

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42:1 – 14, 2023. 2

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3992–4003, 2023. 5, 7

[13] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, 2022. 8

[14] Ming Li, Pan Zhou, Jia-Wei Liu, Jussi Keppo, Min Lin, Shuicheng Yan, and Xiangyu Xu. Instant3d: Instant text-to-3d generation. *arXiv preprint arXiv:2311.08403*, 2023. 3

[15] Weiyu Li, Rui Chen, Xuelin Chen, and Ping Tan. Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. *ArXiv*, abs/2310.02596, 2023. 3

[16] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6526, 2023. 1, 2, 3, 6, 7

[17] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, 2023. 3

[18] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan C. Russell. Editing conditional radiance fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5753–5763, 2021. 3

[19] Zexiang Liu, Yangguang Li, Youtian Lin, Xin Yu, Sida Peng, Yan-Pei Cao, Xiaojuan Qi, Xiaoshui Huang, Ding Liang, and Wanli Ouyang. Unidream: Unifying diffusion priors for relightable text-to-3d generation. *ArXiv*, abs/2312.08754, 2023. 3

[20] Guan Luo, Tianhan Xu, Ying-Tian Liu, Xiao-Xiong Fan, Fang-Lue Zhang, and Song-Hai Zhang. 3d gaussian editing with a single image. In *ACM Multimedia*, 2024. 3

[21] David McAllister, Songwei Ge, Jia-Bin Huang, David W. Jacobs, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Rethinking score distillation as a bridge between image distributions. *ArXiv*, abs/2406.09417, 2024. 3

[22] Luca Medeiros. lang-segment-anything. https://github.com/luca-medeiros/lang-segment-anything, 2024. 4, 5

[23] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *ArXiv*, abs/2212.08751, 2022. 3

[24] Jangho Park, Gihyun Kwon, and Jong Chul Ye. Ed-nerf: Efficient text-guided editing of 3d scene with latent space nerf. In *International Conference on Learning Representations*, 2023. 3

[25] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *ArXiv*, abs/2209.14988, 2022. 2, 3

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen

Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 2, 8

[27] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2021. 1, 6

[28] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and X. Yang. Mvdream: Multi-view diffusion for 3d generation. *ArXiv*, abs/2308.16512, 2023. 1, 2, 4, 5, 6, 7, 8

[29] Christina Tsalicoglou, Fabian Manhardt, Alessio Tonioni, Michael Niemeyer, and Federico Tombari. Textmesh: Generation of realistic 3d meshes from text prompts. *ArXiv*, abs/2304.12439, 2023. 3

[30] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3825–3834, 2021. 3

[31] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Gregory Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12619–12629, 2022. 2, 3

[32] Peihao Wang, Zhiwen Fan, Dejia Xu, Dilin Wang, Sreyas Mohan, Forrest N. Iandola, Rakesh Ranjan, Yilei Li, Qiang Liu, Zhangyang Wang, and Vikas Chandra. Steindreamer: Variance reduction for text-to-3d score distillation via stein identity. *ArXiv*, abs/2401.00604, 2023. 3

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

[34] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Neural Information Processing Systems*, 2016. 3

[35] Jing Wu, Jiawang Bian, Xinghui Li, Guangrun Wang, Ian D Reid, Philip Torr, and Victor Adrian Prisacariu. Gaussctrl: Multi-view consistent text-driven 3d gaussian splatting editing. *ArXiv*, abs/2403.08733, 2024. 3

[36] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6796–6807, 2023. 3, 6, 7, 8

[37] Beichen Zhang, Pan Zhang, Xiao wen Dong, Yuhang Zang, and Jiaqi Wang. Long-clip: Unlocking the long-text capability of clip. *ArXiv*, abs/2403.15378, 2024. 2, 1, 4

[38] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824, 2023. 2, 4, 6, 8

[39] Xiaoyu Zhou, Xingjian Ran, Yajiao Xiong, Jinlin He, Zhiwei Lin, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Gala3d: Towards text-to-3d complex scene generation via layout-guided generative gaussian splatting. *ArXiv*, abs/2402.07207, 2024. 7

Apply Hierarchical-Chain-of-Generation to Complex Attributes Text-to-3D Generation Supplementary Material

Supplementary Material

In this supplementary material, we provide additional information about prompt engineering and conduct additional experiments on the CSP-100 test dataset provided in [5]. We begin with providing prompt engineering details in Sec. 7. Subsequently, we conduct: (1) Additional experiments on CSP-100 test dataset (Sec. 8.1).]; (2) Analysis of Large Language Model (Sec. 8.2); (3) Experiments using Long-CLIP [37] (Sec. 8.3).

7. Prompt Engineering

The prompt which encourages LLMs to create an order of generation is:

"I am now going to use a model to generate text to 3D. This generation is done "inside-to-out", so I need to split the whole sentence into inside-out order as well. Specifically, a sentence contains a subject, instances, and attributes. For example, a yellow dog wearing a blue shirt, a black hat and a red coat is barking. The subject is the yellow dog is barking, and the instances include shirt, hats, and coat, with the corresponding attributes blue, black, and red. What you need to do is, given a prompt, extract the body and the corresponding instance to the attribute. The next step is to carry out stratification. The rule of stratification is that from inside to outside, if there is a relatively obvious occlusion relationship between instances, then it is necessary to expand one layer down. For example, in the above example, the first layer is shirt and hats. The second layer is the coat. So the stratification order is (blue shirt, black hat, EXTEND, red coat) in which the EXTEND means for the next layer. And the sub-prompt of the first layer is: "A yellow dog wearing a blue shirt and a black hat is barking". The sub-prompt of the second layer is: "A yellow dog wearing a red coat is barking." So what you end up returning is to tell me the body, the instances with corresponding attributes the stratification order, and the sub-prompts of each layer. The prompt is: {`input_text`}." In the above prompt, `input_text` is the user's input text.

8. More Experiments

8.1. Experiments on CSP-100 Test Dataset

We conduct additional experiments on the CSP-100 dataset proposed by [5].The CSP-100 dataset encompasses 100 pieces of input texts characterized by considerable compositional complexity, thereby providing a rigorous testbed for evaluating the model's capacity to interpret and generate content conditioned on richly attributed textual descriptions. Examples such as "an orange cat wearing a yellow suit" and "a wooden bird on a stack of green books on a blue chair" illustrate this complexity. The majority of input texts comprise multiple entities or instances, each associated with distinct attributes. Therefore, we perform quantitative comparisons on CSP-100 using BLIP-VQA [9] Score and CLIP Score [8]. We also provide visualizations of selected results. For the reason that the input texts in CSP-100 are relatively simple, the value of original BLIP-VQA tends be in a normal and comparable range for our methods and Progressive3D [5]. the BLIP-VQA Score is used without modification: for each 3D asset, the final score is computed as the product of the part-wise scores, rather than their average.

	Progressive3D [5]	HCoG	HCoG+SD3
BLIP-VQA Score	0.474	0.518	**0.566**
CLIP Score	29.2	29.3	**29.7**

Table 3. **Quantitative comparison on CSP-100 dataset.** HCoG denotes our method based on GALA3D and HCOG+SD3 denotes our method based on GaussianDreamer equipped with Stable Diffusion v3.

The CSP-100 dataset comprises four distinct categories of input texts: Color, Shape, Material, and Composition. The first three categories correspond to individual attribute types, whereas the Composition category involves interactions among multiple objects, each characterized by distinct attributes. As shown in Tab. 3, we perform a **quantitative comparison** on CSP-100 against Progressive3D. Additionally, we evaluate our method on the CSP-100 dataset across different attribute categories in the input text, including **color**, **shape**, **material**, and **composition**. The corresponding BLIP-VQA Scores and CLIP Scores are reported in Tab. 4. The results indicate that our method performs best on complex color attributes, while it achieves extremely competitive results on other attributes of shape, material, and composition.

Additionally, we visualize some results of our method on CSP-100 dataset, which are shown in Fig. 6 and Fig. 7. Fig. 6 shows the results from HCoG based on GALA3D and Fig. 7 shows the results from HCoG based on GaussianDreamer equipped with Stable Diffusion v3. The input text of the CSP-100 dataset is not that complex, so our method can generate relatively high-quality results in most cases.

Figure 6. **Visualization on CSP-100 dataset.** These results are from HCoG based on GALA3D.

	Category	BLIP-VQA	CLIP Score
HCoG	color	**0.653**	**29.7**
	shape	0.438	28.0
	material	0.562	**29.7**
	composition	0.488	29.4
	total	0.518	29.3
HCoG+SD3	color	**0.676**	**30.1**
	shape	0.479	29.2
	material	0.615	29.9
	composition	0.412	29.3
	total	0.566	29.7

Table 4. **Quantitative results of our method on CSP-100 dataset.** We tested the results of different categories of attributes binding, and our method is best at dealing with **color** attributes.

8.2. Reliability Analysis of Large Language Model

Because our method relies on the LLM's ability to generate a near-perfect sequential order, we examined its reliability through the following experiment. Specifically, we selected 300 text prompts with varying part counts n (where higher n indicates greater complexity) and deliberately introduced occlusion between parts. We then used the LLM to produce hierarchical chains. Eleven human experts evaluated the accuracy of the extracted parts (*Num Acc*) and the plausibility of the hierarchical chain (i.e., part ordering based on occlusion) (*Chain Acc*). The experts are also guided to create their own hierarchical chains, which were used to compute the number of inversions in the LLM's output (*Inversion*), as shown in Table 5.

The LLM achieves consistently high part and chain accuracy ($\geq 95\%$), with few inversions, suggesting its robustness in constructing a proper generation order. However, the LLM may fail when n is large. One such failure is illustrated in Figure 8, where an incorrect chain order and missing parts alter the corresponding attributes, leading to misaligned results. The correct generation order should be the scarf first, followed by the cloak. However, the LLM-generated order is reversed, generating the cloak before the

Figure 7. **Visualization on CSP-100 dataset.** These results are from HCoG based on GaussianDreamer equipped with Stable Diffusion v3.

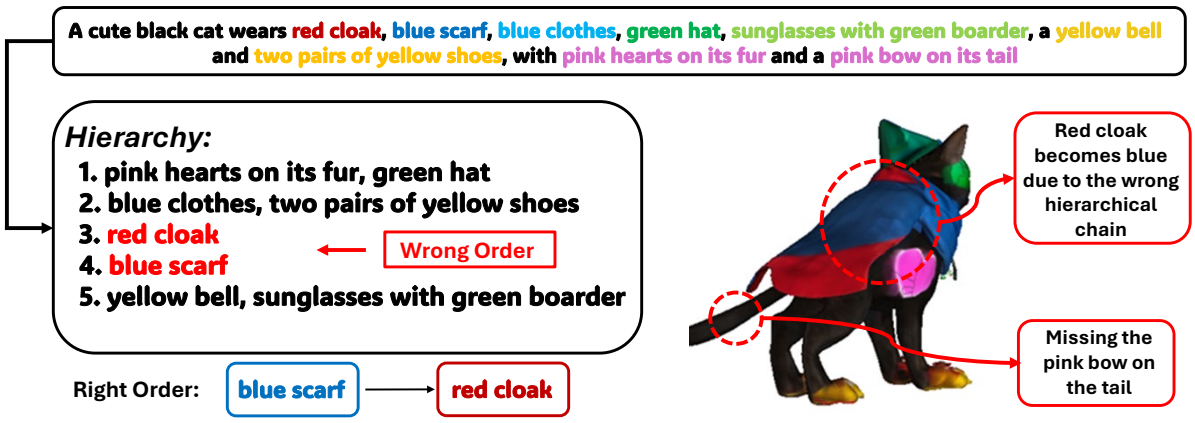


Figure 8. **Failure case for hierarchy chain and results.** There is a wrong order between red cloak and blue scarf, and the pink bow is missing.

scarf. This causes the outer-layer cloak to be influenced by the subsequent scarf generation, resulting in the red cloak being incorrectly rendered as blue. Additionally, the pink bow on the cat's tail is missing. Under similarly complex textual inputs, such failure cases are likely to occur with the LLM.

	$1 \leq n \leq 5$	$6 \leq n \leq 10$	$n \geq 11$
Num Acc ↑	**1.00**	0.99	0.98
Chain Acc ↑	**0.99**	0.97	0.95
Inversion ↓	**0.95**	2.00	3.65

Table 5. **LLM ability for the hierarchy of parts and chain orders.** In most cases, LLM can achieve satisfactory performance.

8.3. Long-CLIP

Figure 9. **The results of diffusion model with Long-CLIP in terms of complex input text.** Diffusion model with Long-CLIP is still unable to solve the problem of complex attributes binding.

We testify other approaches to handle the challenge of long and complex input text, one potential solution is to employ a CLIP variant capable of handling such inputs, such as Long-CLIP [37]. We therefore utilize Long-CLIP [37] to generate images from long and complex textual descriptions, as illustrated in Fig. 9. However, experimental results show that even with Long-CLIP, handling complex attribute-rich text remains challenging in 2D image generation. Since 3D generation builds upon 2D representations, directly applying Long-CLIP to 3D generation tasks yields limited improvement.