From Geometry to Texture: A Hierarchical Framework for Efficient Text-to-3D Generation

Yangguang Li[1], **Zehuan Huang**[2], **Feng Liang**[3], **Bin Huang**[1], **Qinghong Sun**[1]
Xihui Liu[4], **Lu Sheng**[2], **Wanli Ouyang**[5], **Jing Shao**[1,5]

[1] SenseTime, [2] Beihang University, [3] The University of Texas at Austin
[4] The University of Hong Kong, [5] Shanghai AI Laboratory
{liyangguang, huangbin1, sunqinghong1, shaojing}@sensetime.com
{huangzehuan,lsheng}@buaa.edu.cn, jeffliang@utexas.edu
xihuiliu@eee.hku.hk, ouyangwanli@pjlab.org.cn

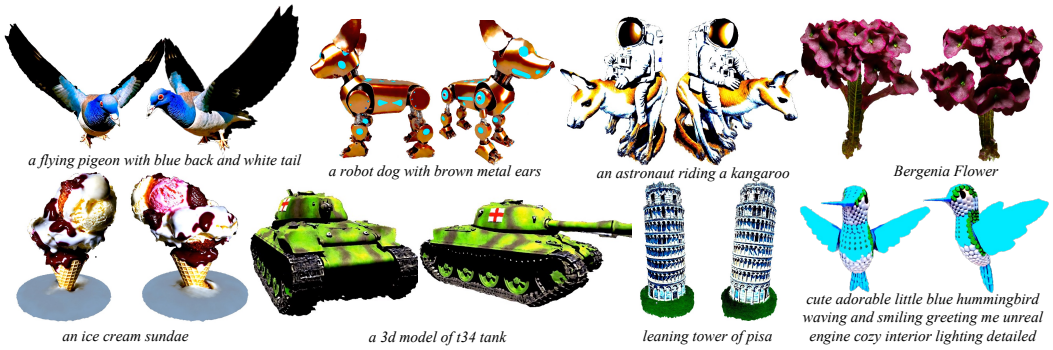


Figure 1: Our Hi3D model efficiently creates high-quality 3D models from various text prompts. These prompts can include descriptions of motion (*e.g.*, *flying pigeon*, *astronaut riding*), materials (*e.g.*, *brown metal*), and specific object identities (*e.g.*, *T34 tank*, *Pisa Tower*).

Abstract

Text-to-3D generation has achieved remarkable success by leveraging pre-trained text-to-image diffusion models to optimize Neural Radiance Fields (NeRFs). However, these approaches usually suffer from slow convergence since the optimization of 3D geometry and RGB texture is heavily intertwined because of the volume rendering process, and the 2D supervision provided by 2D diffusion models usually explains multiple 3D geometries. Therefore, we introduce **Hi3D**: a **Hi**erarchical framework for efficient text-to-**3D** generation that disentangles the optimization of geometry and texture. Specifically, our approach first generates 3D geometry and then progressively generates fine-grained textures. Unlike prior works which optimize the 3D geometry from scratch, we start with a sparse 3D point cloud which can be instantly obtained via a pre-trained 3D generative model, such as Point-E. The generated point cloud initializes the geometric occupancy and provides shape guidance to accelerate training. To further enhance texture, we refine the RGB outputs of the NeRF model and then optimize the reconstructed surfaces for finer details, in a cascading manner. Through experiments, our proposed **Hi3D** is 6.6 $\times$ faster than prior works (*e.g.*, Magic3D) under comparable generation quality, and can perform image-driven controllable 3D model generation, with fewer distortions caused by ambiguous geometry clues from 2D supervisions.

Technical Report 2023.04.

1 Introduction

Producing high-quality 3D content with rich geometric and texture details is crucial for many fields, such as gaming, film, and the AR/VR industries. However, this task demands an immense amount of workload for specialists in related areas. The impressive success of large-scale text-to-image diffusion models [1, 2, 3] in image content generation has inspired researchers to explore generating 3D assets from textual descriptions, potentially lowering 3D content creation costs. One major bottleneck of directly training 3D diffusion models is the lack of large-scale text-3D paired data. The largest public text-3D dataset Objaverse [4] contains only 800K text-3D pairs while image-text datasets, such as Laion-5B [5], contain billions of text-image pairs.

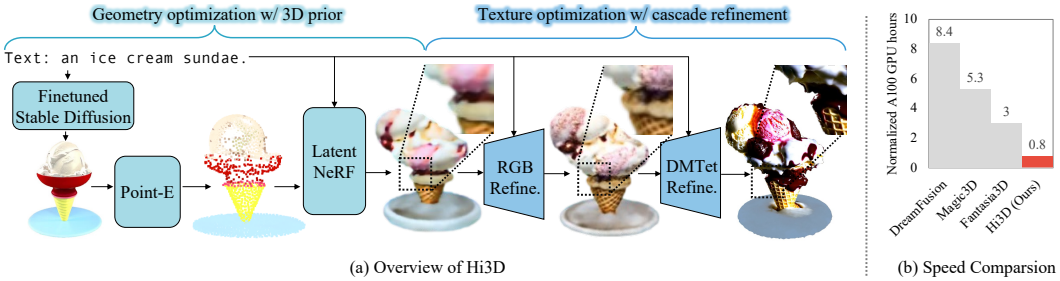
To bypass this hurdle, researchers propose to leverage pre-trained text-to-image diffusion models for 3D content generation without any 3D data. Pioneering work DreamFusion [6] optimizes Neural Radiance Fields (NeRFs) [7] by back-propgating the supervision from text-to-image diffusion models. More specifically, given an input text prompt, the Neural Radiance Field (NeRF) is optimized through Score Distillation Sampling (SDS) such that its 2D renderings from random angles achieve a low loss. Although DreamFusion shows great potential, it is very slow (6 TPUv2 hours per scene) and can only generate low-resolution (64×64) content. We argue the inefficiency majorly lies in the fact that it generates 3D geometry and RGB texture simultaneously via optimizing an inefficient entangled volume rendering process, and the 2D supervision provided by 2D diffusion models may explain multiple 3D geometries. This problem becomes more severe if NeRF goes into latent space [8] where fine-grained textures are more challenging to represent.

This paper introduces **Hi3D**: a **Hi**erarchical framework for efficient text-to-**3D** generation, as shown in Fig.2(a). Rather than concurrently optimizing 3D geometry and texture, Hi3D initially concentrates on generating 3D geometry and subsequently incorporates detailed texture information in a progressive manner. We show that not only does our disentangled design yield better generation quality, but it is also more efficient to train. Furthermore, it is capable of image-driven controllable 3D generation and can resolve 3D geometric distortions caused by ambiguous geometric cues from 2D supervision.

In particular, we first generate the 3D geometry with Latent-NeRF [8] framework. Unlike previous works starting from scratch [6, 9] or using limited sketch shapes [8], we employ a case-specific coarse point cloud which can be instantly obtained via a 3D generative model, Point-E [10]. This point cloud offers a rich 3D geometry prior which can be exploited to accelerate NeRF training. However, challenges arise in balancing prior utilization and NeRF expressiveness. Our goal is to align NeRF output with the point cloud's basic skeleton while preserving its freedom to express the geometry of the text prompt. To achieve this, we initialize NeRF's geometric occupancy with generated point clouds and constrain its training using adaptive shape guidance. More specifically, we encourage NeRF's occupancy to match the coarse point cloud's surface distribution while gradually reducing constraint weight near the point cloud's surface, thereby preserving NeRF's expressiveness for geometry modeling.

The initial stage of Latent-NeRF successfully generates geometric structures. However, it lacks texture details due to its optimization in the latent space. To address this, we introduce a cascading refinement process to enhance texture. Firstly, we perform RGB refinement [8], allowing the model to capture texture details and color information while preserving overall geometry. Next, we optimize finer reconstructed surfaces using DMTet [11], which employs the Marching Tetrahedra (MT) technique for finer surface optimization. This stage refines fine-grained texture details and captures sharp features missed in previous steps. We find that the hierarchical refinement approach reduces texture learning difficulty and results in higher-quality texture expression.

Through extensive experiments, the proposed Hi3D presents a unique ability from three aspects: (1) Efficient: Hi3D is currently the fastest NeRF-based text-to-3D pipeline, which only needs 0.8 GPU hours to achieve the comparable performance with Magic3D [9]. As shown in Fig.2(b), Hi3D is 10.5 $\times$ faster than DreamFusion and 6.6 $\times$ faster than Magic3D, and 3.75 $\times$ faster than Fantasia3D. (2) Controllable: Current research on text-to-3D generation mainly focuses on the effect rather than its controllability. Hi3D, alternatively, provides controllable capacity on constraining NeRF to obtain a geometric shape similar to the point cloud. Specifically, given a certain text prompt, we can select the image generated by the prompt to guide the generation of the desired point cloud, thereby controlling the shape of the target object generated by NeRF. (3) Geometric stable: Existing schemes are prone to unstable geometric optimization due to ambiguous geometry interpretation from 2D supervisions,



(a) Overview of Hi3D (b) Speed Comparsion

Figure 2: (a). Given a text prompt, the proposed Hi3D first employs a 3D generative model, Point-E, to generate a rough 3D point cloud. Then, Hi3D optimizes a Latent-NeRF to derive the 3D geometry, leveraging the 3D prior. Finally, Hi3D progressively enhances the texture through a cascading refinement process, which consists of RGB and DMTet refinement. (b). Our Hi3D achieves the best efficiency[2]when compared with other text-to-3D frameworks.

leading to distortions in NeRF-trained geometries. Hi3D can maintain relatively good geometric stability through point cloud constraints.

In conclusion, our contributions are summarized as follows: (1) We propose a hierarchical framework for efficient text-to-3D generation, which disentangles the optimization of geometry and texture. (2) We accelerate the geometry optimization with 3D prior from generated 3D point clouds. We propose point cloud initialization and adaptive shape guidance to balance 3D prior utilization and NeRF expressiveness. (3) We introduce a cascaded approach for texture enhancement, employing RGB refinement for color information and DMTet for finer surface optimization. (4) Experiments prove that Hi3D achieves excellent text-to-3D generation speed, which is 6.6 $\times$ faster than Magic3D with comparable performance. Moreover, Hi3D can effectively control 3D content generation by selecting images and can well handle the situation where the optimization direction is unstable.

2 Related Work

2.1 3D generation with 3D diffusion models

Text-to-3D generation first explored the concept of 3D diffusion models, similar to text-to-image diffusion models. A prime example of such work is SDFusion [12], which employs a pipeline similar to the stable diffusion model. It encodes the original input into the latent space and performs 3D diffusion operations in this latent space based on text or image conditions. This generates corresponding features of the 3D model, which is then decoded into a 3D model. Works like [13, 14, 15, 16, 17, 18] are similar to SDFusion, investigating the impact of different 3D diffusion methods on specific small 3D datasets. Despite the success of these methods, their applicability is limited by training resources and data scale constraints, with experiments primarily conducted on the ShapeNet [19] dataset. The first exploration of large-scale training was conducted by Point-E [10], which utilized several million text-point data to train a 3D point cloud diffusion model, generating corresponding 3D shapes from text or images. Shap-E [20] further optimized Point-E's approach, combining diffusion and NeRF to generate superior 3D models. Additionally, 3DGen [21] used Objaverse [4] data to investigate the end-to-end text-to-3D diffusion model generated simultaneously by SDF and color, yielding impressive results. However, all these schemes are heavily reliant on large-scale 3D datasets. The collection, storage, and training of such large-scale 3D datasets pose significant challenges.

2.2 3D generation with 2D diffusion models

Pioneering DreamFusion [6] proposes an alternative approach to leverage a 2D diffusion model to optimize NeRF [7, 22]. Our methodology, Hi3D, also falls into this category. Subsequent work, such as Magic3D [9] improves training efficiency with a faster NeRF model Instant-NGP [23] and a DMTet refinement [11] to increase the resolution. Latent-NeRF [8] proposes accelerating training

[2]For a fair comparison, the speed is normalized to A100 GPU hours.

by optimizing NeRF in the latent space. 3DFuse [24] incorporates 3D awareness into 2D diffusion models by using a coarse 3D structure provided by 3D model. Fantasia3D [25] shares the same motivation by disentangling the geometry and texture. Our Hi3D framework is different from these existing methodologies in two key ways: (1) Instead of starting from scratch [6, 9] or relying on limited sketch shapes [8, 25], we utilize the 3D generative model, Point-E, to provide a 3D prior, so as to accelerate geometry optimization. (2) We introduce a cascade texture refinement, as opposed to no refinement [6, 24] or a single refinement [8, 9, 25], so as to enhance texture. In addition to enhanced training efficiency, our Hi3D framework also provides a unique advantage in terms of control. It allows for more deliberate 3D object generation with the incorporation of image guidance, a feature that is not supported by the aforementioned methods.

3 Methods

In this section, we first revisit DreamFusion [6] and Latent-NeRF [8] to figure out their limitations (Sec. 3.1). Then, we introduce our **Hi**erarchical text-to-**3D** framework **Hi3D** which disentangles the optimization of geometry and texture. Hi3D first focuses on generating the 3D geometry (Sec. 3.2), then progressively adds detailed texture information (Sec. 3.3). Finally, we demonstrate Hi3D's capability to produce more controllable 3D objects using image guidance (Sec.3.4).

3.1 Revisiting DreamFusion and Latent-NeRF

DreamFusion. Pioneering DreamFusion [6] proposes to leverage pre-trained text-to-image diffusion models Imagen [2] to generate 3D objects, without 3D training data. The key idea is to train a 3D Neural Radiance Field (NeRF) [7, 22] that can provide seamless and text-aligned views from different camera poses. The 3D NeRF is optimized by back-projecting the supervision from the 2D diffusion model. Formally, we denote the NeRF model as $x = g(\theta)$, where θ is coordinate-based MLP representing the 3D volume, $g(\cdot)$ is the renderer, and x is the generated view at a given camera pose. Given a random rendered view x, DreamFusion adds Gaussian noises ϵ on the view and reconstructs it with the frozen diffusion model ϕ to predict the injected noises $\hat{\epsilon}_\phi(x_t; y, t)$, where t is the timestep representing noisy level, y is the text condition, and x_t is the noised image. Subtracting the predicted noise, $\hat{\epsilon} - \epsilon$, offers a signal for aligning the rendered view x with the text input y, as perceived by the diffusion model. DreamFusion updates the NeRF MLP parameters by backpropagating the gradient through the rendering process using Score Distillation Sampling (SDS), as depicted in Eq. 1.

$$\nabla_\theta L_{SDS}(\phi, g(\theta)) = \mathbb{E}_{t,\epsilon}\left[w(t)(\hat{\epsilon}_\phi(x_t; y, t) - \epsilon)\frac{\partial_x}{\partial_\theta}\right] \tag{1}$$

Here, $w(t)$ is a weighting function that depends on the timestep t. For 2D diffusion models, unlike DreamFusion [6], Magic3D [9] using pixel-based models, such as Imagen [2] or eDiff-I [26], we use latent-based Stable-Diffusion [3] because of its availability and high efficiency. For NeRF model, we use Instant-NGP [23] instead of DreamFusion's Mip-NeRF360 [22] for efficiency consideration. Although DreamFusion shows great potential, when it comes to surface texture recovery, NeRF modeling proves to be less effective [27, 28].

Latent-NeRF. For the best pursuit of efficiency, we use Latent-NeRF [8] and 2D latent Stable Diffusion [3]. The entire process, including view rendering, NeRF geometry generation, and SDS optimization are all operated in latent space. Formally, similar to DreamFusion, the difference is to replace the image level diffusion with latent level diffusion. Same as in Eq. 1, replace x_t to z_t.

Owing to the vast flexibility in NeRF geometric shapes, Latent-NeRF [8] incorporates sketch-shape guidance to constrain the geometry and speed up training. However, acquiring coarse sketch shapes can be challenging and limited due to the smaller amount of 3D data, as well as the fact that most of the data are private or fee-based. Additionally, we find that the winding number [29] operation on the sketch-shape is considerably slow, which could negatively impact the overall training speed.

3.2 Geometry optimization

Point-E as 3D prior. We propose to use 3D generative models, specifically Point-E[10], to offer a 3D geometry prior to accelerating NeRF training. Unless otherwise specified, we use image instead of text guidance to help Point-E to generate high-quality point cloud. We use Stable Diffusion [3] to

generate images instead of Point-E's GLIDE model [30] which is not publicly available. However, we observe a significant domain gap between the generated images of Stable Diffusion and appropriate Point-E input images. That is, Stable Diffusion tends to generate images with various object sizes, styles and complex backgrounds, whereas Point-E requires images with centered objects and clean backgrounds. To address this issue, we propose to finetune the Stable Diffusion model with the collected mini dataset provided by Point-E. More specifically, we use 153 text-image pairs from Point-E's official repo[3]. Due to the limited size of the dataset, we propose employing the Dreambooth method [31] to fine-tune the model.

Formally, given a text prompt y, we first employ the fine-tuned Stable Diffusion model to synthesize a view image I. Next, we input I into Point-E's point cloud diffusion process to generate the point cloud P. Users can also provide the view image, enabling more controllable point cloud generation, as discussed in Sec. 3.4. Using the obtained point cloud P, we initialize NeRF's geometric occupancy and provide shape guidance for the ongoing NeRF training, as explained in the following section.

Geometry Occupancy Initialization. We adopt the INGeo approach [32] to initialize NeRF's geometric occupancy using coarse point clouds derived from Point-E. Specifically, as depicted in the initialization part of Fig.3, we carry out three distinct operations of INGeo, namely "*Density Scaling*", "*Point-Cloud Splatting*", and "*Updating Occupancy Grids*" on the point cloud. Its purpose is to use the dense initialization point cloud with consistent shape as the geometric prior and use the density after scaling as the clue after the NeRF's MLP output to highlight the shape after initialization. However, we discover that training NeRF solely through SDS results in a high degree of freedom for representing the target's geometric shape, leading to a mismatch with initialized point cloud shape and, consequently, suboptimal performance. This is also found in INGeo that, without applying further constraints during training, the optimization process would diverge, making the initialization ineffective. To address this issue, we introduce point cloud shape guidance, which leverages the geometric constraints of coarse point clouds to ensure the object shape generated by NeRF aligns with the shape distribution of the initial point cloud, as detailed in the following section.

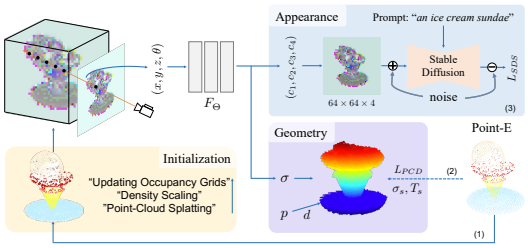


Figure 3: Geometry optimization process. (1) Initialization: The point cloud from Point-E is used to initialize NeRF's geometric occupancy. (2) Geometry: The point cloud imposes shape guidance on the results of NeRF. (3) Appearance: NeRF is optimized using the pre-trained 2D model by SDS.

Adaptive Shape Guidance. We impose constraints on the density distribution of NeRF's MLP output to align with the 3D point cloud geometry. This ensures that the target geometry trained in NeRF is similar to the initial point cloud, preventing divergent optimization within NeRF. However, it is also crucial to maintain a certain shape expression space for NeRF, allowing it to further optimize over the coarse point cloud. To accomplish this, we propose adaptive shape guidance that changes constraints for different grids in NeRF. Specifically, we gradually reduce the constraint weights near the point cloud surface, preserving NeRF's spatial shape expression capabilities. Formally, as shown in Eq. 2.

$$L_{PCD} = CE(\alpha_{NeRF}(p), \alpha_{PCD}(p)) \cdot (1 - e^{-\frac{d}{2\sigma_s^2}}) \tag{2}$$

Here, p represents the entire point set within the NeRF space, while d denotes the minimum distance from the sampling point to the point cloud in the NeRF space. The hyperparameter σ_s signifies the point cloud constraint's looseness, with a lower σ_s implying tighter constraints. As shown in geomertry part of Fig.3, given the point cloud surface distance threshold T_s, when $d < T_s$, the sampling point in the NeRF space is considered to be near the point cloud surface, indicating 1, and the remaining points are 0. Thus, the distribution of spatial sampling points near and far from the geometric surface is defined. A point cloud geometry constraint loss is established by calculating the cross-entropy between this distribution and the NeRF space occupancy distribution. This loss implies that occupancy constraints should be stronger farther from the point cloud surface and weaker closer

[3]https://github.com/openai/point-e

to it. As a result, the influence of low-resolution point clouds on the smoothness of the final geometry is minimized. In the setting of point cloud-guided geometry learning, we employ the following loss:

$$L = L_{SDS} + \lambda_{PCD} L_{PCD} \tag{3}$$

In this equation, L_{SDS} is the Score-Distillation loss mentioned in Section 3.1. λ_{PCD} is a weight assigned to L_{PCD} which is typically quite small. This allows for improved scene perception during the learning process, while also converging towards the specified geometry.

3.3 Texture Refinement

We additionally improve the texture through a cascading refinement process. During texture refinement, we solely optimize the texture while maintaining the 3D geometry unchanged. It is important to note that we employ the same Stable Diffusion model for both geometry and texture optimization; however, these optimizations occur in distinct spaces—geometry takes place in the latent space, whereas texture occurs in the pixel space.

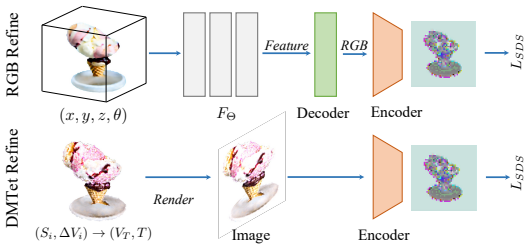


Figure 4: Texture refinement process. RGB Refine: the output of F_Θ(MLP) first transferred to RGB space for encoding to obtain features and then performed diffusion. DMTet Refine: render the image from the mesh, and perform the diffusion operation after encoding.

RGB Refinement. Similar to the process such as DreamFusion [6], as shown in the first row of Fig.4, the RGB representation of a certain view is obtained by decoder after rendering from (x, y, z, θ). And use the image encoder to convert the RGB space into latent space. Then perform the same diffusion process as the geometry stage to optimize the texture representation of the target in NeRF. The MLP (F_Θ) of NeRF in the RGB refine stage is initialized with the weight of the MLP in the geometry stage.

DMTet Refinement. While the RGB space provides a good representation of texture, it lacks the capacity to capture finer details. Following Magic3D [9], we incorporate DMTet [11] optimization to enhance the quality of texture expression. We initiate DMTet training with the 3D shape represented by the NeRF. We employ a deformable tetrahedral grid (V_T, T) for representation, where V_T denotes the vertices in the grid T. Each vertex $v_i \in V_T \subset R^3$ comprises a signed distance field (SDF) value $s_i \in R$ and a deformation $\triangle v_i \in R^3$ from its initial canonical coordinate [9]. DMTet refinement is carried out in three steps. Firstly, we use a differentiable marching tetrahedra algorithm [11] to extract the surface mesh from the SDF. Secondly, we employ a differentiable rasterizer [33] to render the extracted surface mesh into a high-resolution image. Finally, we use SDS gradients to optimize s_i and $\triangle v_i$ for each v_i. The DMTet process can be seen in the second row of Fig.4.

3.4 Controllable 3D generation with image guidance

Due to the inherent ambiguity of text prompts, current 3D generation pipelines [6, 9, 25] can yield considerably different results even when using the same prompt. However, with the integration of Point-E, we can provide image guidance to the point cloud diffusion model in order to obtain an image-specific point cloud, which subsequently guides the generation of the final 3D object. More interestingly, our model is capable to work with foundation vision models, such as SAM [34] and BLIP-2 [35], to generate 3D objects directly from 2D images. See Sec.4.2 and supplementary details for more details.

4 Experiments

4.1 Implementation Details.

Model selection. We use the Stable Diffusion model v2-1 [3] as our 2D text-to-image diffusion model. To provide better image guidance to Point-E, we fine-tune the Stable Diffusion model using 153 text-image examples derived from Point-E's official repository. It is worth noting that fine-tuned

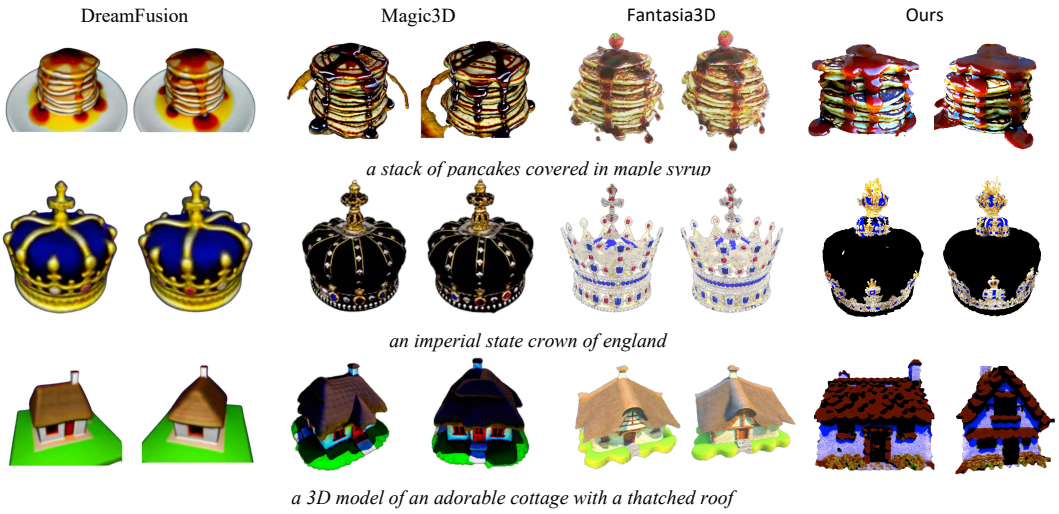


Figure 5: Qualitative comparsion with other text-to-3D frameworks, DreamFusion, Magic3D, and Fantasia3D. Our Hi3D model reveals comparable performance with existing methods.

Stable Diffusion model is solely used to generate images for Point-E. All other processes, including Latent-NeRF and texture refinement, utilize the default Stable Diffusion model. We employ the largest image-to-point-cloud Point-E model with 1 billion parameters, to generate point clouds. The pose of the point cloud is further adjusted so that the Z axis is perpendicular to the horizontal plane. While we have experimented with Point-E's text-to-point-cloud model and other smaller image-to-point-cloud modes, we found that the largest model consistently delivered superior performance.

Geometry optimization. We utilize Latent-NeRF [8] as the initial step to optimize the 3D geometry. This involves initializing the geometry occupancy and incorporating adaptive shape guidance during the training phase. For the training, we use a latent size of 64×64. The guiding point cloud is normalized to be within a sphere of radius 0.7. For the guidance loss L_{PCD}, we use the following parameters: a surface distance threshold T_s of 0.04, a constraint looseness σ_s of 0.2, and a weight for the point cloud constraint λ_{PCD} of 5e-7. We conduct this training phase on a single A100 GPU, completing 3K iterations in approximately 13 minutes.

Texture refinement. Once we have obtained the basic geometry, we proceed to the RGB refinement stage, initializing the weights from the Latent-NeRF stage. We adhere to the training hyperparameters used in Latent-NeRF. The training resolution for this RGB refinement stage is set at 512×512. This phase involves 2K iterations of training on a single A100 GPU, which approximately takes around 15 minutes. Following the RGB refinement, we proceed with DMTet refinement. We begin by extracting the 3D mesh from the RGB model to serve as the initial input. The "density threshold" applied during the extraction of the mesh is set to 1.0. The "default theta" and "default fovy" are set at 80 and 70 respectively. The training resolution for this stage is 512×512. This process involves 3K iterations of training on a single A100 GPU, which approximately takes around 17 minutes. During the testing stage, we set the "bg radius" to zero to generate a 3D model against a white background.

4.2 Comparison to other text-to-3D methods

Qualitative results. We perform a qualitative comparison of our method, Hi3D, with existing text-to-3D generation methods including DreamFusion [6], Magic3D [9], and Fantasia3D [25]. To ensure fair comparisons, we use identical text prompts from Fantasia3D for all methods. The results show that Hi3D produces 3D models of comparable quality to existing methods. It's noteworthy that our method requires significantly less time to generate an object, only 45 minutes, which is approximately 27.0% of the time taken by Fantasia3D and 15.0% of

Table 1: User study comparsion.

Method	User Preference(%)
Stable-DreamFusion [36]	0
Stable-Magic3D [36]	17.4
Hi3D (Ours)	**82.6**

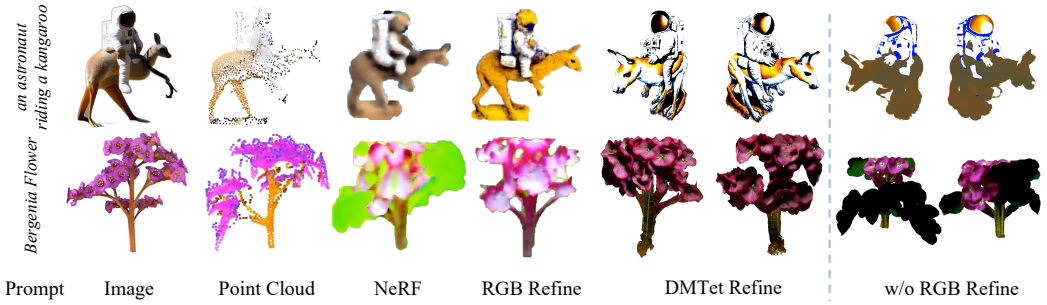


Figure 7: Left: Intermediate result of each step in Hi3D. Right: Result without RGB refinment: directly from NeRF stage to DMTet stage.

that by Magic3D. This underscores Hi3D's efficiency in generating high-quality 3D objects. We also showcase more examples in Fig.1.

Quantitative results. To quantitatively compare our method with existing ones, we utilize reimplementations of DreamFusion [31] and Magic3D [9] based on Stable-DreamFusion [36], as their official implementations are not publicly accessible. We first select 86 text prompts from DremFusion, Magic3D and Objaverse and use these prompts to generate 3D objects using various methods. It's worth noting that existing methods, such as Stable-DreamFusion, can be quite unstable, often resulting in training collapse or excessively noisy objects. To circumvent this, we retry the same prompt up to three times. If after three attempts, a valid object is not generated, we consider this a failure for the method on that particular prompt. Following the generation, two independent annotators are tasked with selecting the best object from among all the methods we should considering. Their selections are only recorded when both annotators agree. In cases of disagreement, a third annotator is brought in to make the final assessment. As presented in Tab.1, our method consistently outperforms existing other methods, achieving an overall favorability of 82.6%. We'd like to note that quantitative comparisons with the official models of DreamFusion and Magic3D are not conducted due to the unavailability of their implementations.

Table 2: Training cost (normalized A100 GPU hours) comparison between our Hi3D and other methods.

Methods	Overall	NeRF	RGB Refine	DMTet Refine
DreamFusion [6]	8.4	8.4	-	-
Magic3D [9]	5.3	2	-	3.3
Fantasia3D [25]	3	1.7	-	1.3
Hi3D (Ours)	**0.8**	**0.2**	**0.3**	**0.3**

Cost breakdown. We also provide a detailed analysis of the cost breakdown for our method compared to other existing methods. Because different methods utilize different hardware, we use training cost in their papers and convert it to the equivalent running time on a single A100 GPU. More details about conversion can be found in supplementary details. We divide the overall cost into three components: NeRF optimization, RGB refinement, and DMTet refinement, as shown in Tab.2. For methods that do not have a certain stage, we leave that field blank. Our model, thanks to the introduction of 3D priors and cascade refinement, achieve a faster speed even with more stages, as can be seen from the comparison.

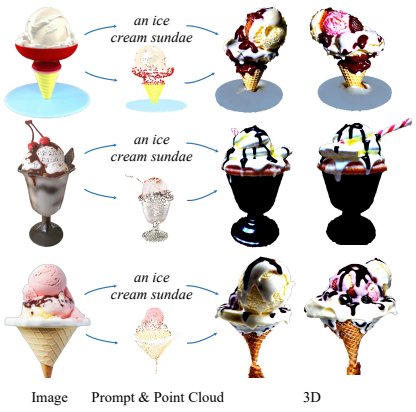


Figure 6: Image-driven 3D controllability generation process.

Controllable 3D generation with image guidance. We demonstrate the controllability of our 3D generation. As illustrated in Fig.6, given the same text prompt but different images, Hi3D can generate distinct 3D results that correspond to the content of the images. This provides the user with more controllability over the final object generation. Moreover, our model can work with foundation vision models, such as SAM [34] and BLIP-2 [35], to generate 3D objects directly from 2D images. More results can be found in the supplementary details.

5 Ablation Study

Intermediate results in Hi3D. As Hi3D is a multi-stage hierarchical pipeline, we show the intermediate result of each step in Fig.7. Starting with text prompts, Hi3D gets point clouds with similar semantic to prompt and consistent content to image. The NeRF stage, benefiting from the 3D prior provided by the point cloud, effectively produces 3D geometry that closely resembles the input point cloud. From the RGB refinement stage to the DMTet refinement stage, there is a noticeable increase in texture detail. While the RGB refinement primarily enhances color information, the DMTet stage further refines the model by adding more fine-grained texture details.

Ablation on texture refinement. We further conducted an ablation study to examine the effect of excluding the RGB refinement stage. As seen in the last column of Fig.7, the models generated using only the DMTet refinement process without preceding RGB refinement, underperform compared to those subjected to both stages. This observation underscores the significance of the RGB refinement phase in our model.

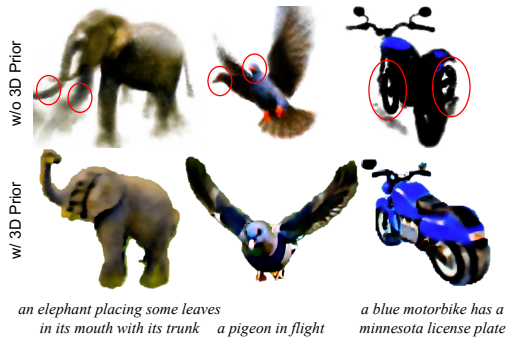


Figure 8: Effectiveness of initialization and shape guidance.

Ablation on 3D prior. Training a 3D NeRF using SDS can be unstable because the 2D supervision from 2D diffusion models can often correspond to multiple 3D geometries. This phenomenon is exemplified in the first row of Fig.8, where the generated objects may develop along two different geometric optimization directions, a situation we refer to as the "dual-head" problem. Our Hi3D method tackles this challenge by incorporating a point cloud as a 3D prior, significantly enhancing the training stability. This improvement is illustrated in the second row of Fig.8.

6 Limitation

While Hi3D demonstrates impressive efficiency in producing high-quality 3D objects from a variety of text prompts, it does have a few limitations. (1) The quality of point clouds generated by Point-E greatly influences subsequent stages and the final 3D outcomes. The capability to obtain a valid 3D object starting with a suboptimal point cloud remains a point of uncertainty. (2) While coarse point clouds exhibit excellent control over geometry, their ability to manage color and texture is limited. This is due to the fact that texture requires dense guidance, which the point cloud cannot sufficiently provide. The current model might produce textures that are not entirely consistent with the original point cloud's color, leading to possible discrepancies in the final 3D object's appearance.

7 Conclusion

In this work, we introduce an efficient hierarchical framework for the progressive optimization of 3D objects' geometry and texture. Our approach begins with the use of a sparse point cloud, generated by Point-E, as a prior to initialize the geometric space occupancy of Latent-NeRF. We propose adaptive shape guidance to balance the use of 3D prior and the expressive capability of NeRF. Upon obtaining a valid geometry, we enrich it with precise color information through RGB refinement and optimize the surface for finer details using DMTet. Our Hi3D model demonstrates significant improvements over existing methods in terms of efficiency, controllability, and geometric stability.

8 Acknowledgments

We would like to thank Chaojian Li from Georgia Institute of Technology for his advice and help on the application of INGeo in Hi3D. Also thanks to Matthieu Lin of Tsinghua University for the help of INGeo's application in Hi3D.

References

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

[2] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[3] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.

[5] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

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

[7] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[8] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. *arXiv preprint arXiv:2211.07600*, 2022.

[9] 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. *arXiv preprint arXiv:2211.10440*, 2022.

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

[11] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.

[12] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander Schwing, and Liangyan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. *arXiv preprint arXiv:2212.04493*, 2022.

[13] Zhen Liu, Yao Feng, Michael J Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. Meshdiffusion: Score-based generative 3d mesh modeling. *arXiv preprint arXiv:2303.08133*, 2023.

[14] Biao Zhang, Matthias Nießner, and Peter Wonka. 3dilg: Irregular latent grids for 3d generative modeling. *arXiv preprint arXiv:2205.13914*, 2022.

[15] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[16] Aditya Sanghi, Rao Fu, Vivian Liu, Karl Willis, Hooman Shayani, Amir Hosein Khasahmadi, Srinath Sridhar, and Daniel Ritchie. Textcraft: Zero-shot generation of high-fidelity and diverse shapes from text. *arXiv preprint arXiv:2211.01427*, 2022.

[17] Rao Fu, Xiao Zhan, Yiwen Chen, Daniel Ritchie, and Srinath Sridhar. Shapecrafter: A recursive text-conditioned 3d shape generation model. *arXiv preprint arXiv:2207.09446*, 2022.

[18] Muheng Li, Yueqi Duan, Jie Zhou, and Jiwen Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. *arXiv preprint arXiv:2212.03293*, 2022.

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

[20] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions, 2023.

[21] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023.

[22] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[23] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[24] Junyoung Seo, Wooseok Jang, Min-Seop Kwak, Jaehoon Ko, Hyeonsu Kim, Junho Kim, Jin-Hwa Kim, Jiyoung Lee, and Seungryong Kim. Let 2d diffusion model know 3d-consistency for robust text-to-3d generation. *arXiv preprint arXiv:2303.07937*, 2023.

[25] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*, 2023.

[26] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

[27] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[28] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021.

[29] Gavin Barill, Neil G Dickson, Ryan Schmidt, David IW Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[30] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[31] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *arXiv preprint arXiv:2208.12242*, 2022.

[32] Chaojian Li, Bichen Wu, Albert Pumarola, Peizhao Zhang, Yingyan Lin, and Peter Vajda. Ingeo: Accelerating i nstant n eural scene reconstruction with noisy geo metry priors. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pages 686–694. Springer, 2023.

[33] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8280–8290, 2022.

[34] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[35] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[36] Jiaxiang Tang. Stable-dreamfusion: Text-to-3d with stable-diffusion, 2022. https://github.com/ashawkey/stable-dreamfusion.

[37] 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 Girshick. Segment anything. *arXiv:2304.02643*, 2023.

A Pseudo code

We show our Hi3D pipeline in the pseudo-code as follows. We will open-source our codes and models in the future.

Algorithm 1 Hi3D pipeline

Input: X, θ, F_Θ, $UNet$, X_{PCD}, λ_{PCD}, T_s, δ # X, θ means position and direction sampled in NeRF. F_Θ is an MLP which takes X, θ as input and outputs latent feature C and density σ. X_{PCD} is the distribution of points in the prior point cloud. λ_{PCD} is the weighting function over loss L_{PCD}. T_s is threshold for judging whether points are on the surface of the prior point cloud. δ is used to compute the occupancy (α) of NeRF.

function BATCH-UPDATING(X, θ, F_Θ)
    # Initialization.
    $X^{'}_{PCD} \leftarrow$ SPLATTING(X_{PCD})
    DENSITY-SCALING(X) $\leftarrow X^{'}_{PCD}$

    # Get Latent Feature of View Rendered From NeRF.
    $C, \sigma \leftarrow F_\Theta(X, \theta)$

    # Calculate The Losses.
    $L_{SDS} \leftarrow$ SDS-LOSS(C)
    $L_{PCD} \leftarrow$ PCD-LOSS(X, σ)
    $L \leftarrow L_{SDS} + \lambda_{PCD} L_{PCD}$

    # Update The Network Initialization.
    $F_\Theta \leftarrow$ BACKWARD-UPDATE(F_Θ, L)
    $X^{'}_{PCD} \leftarrow$ UPDATING($X^{'}_{PCD}$)
end function

function SDS-LOSS(C)
    $x_t, \epsilon \leftarrow$ ADD-NOISE(C, t)
    $\tilde{\epsilon} \leftarrow$ PREDICT-NOISE($UNet, x_t, t$)
    $L_{SDS} \leftarrow \tilde{\epsilon} - \epsilon$
    return L_{SDS}
end function

function PCD-LOSS(X, σ)
    # Get Sampling Points Close to Surface of Point Cloud. d denotes the minimum distance from the sampling point to the point cloud in the NeRF space.
    $d \leftarrow$ MIN-DIST(X, X_{PCD})
    $surface\_inds \leftarrow d < T_s$

    # Loss between Occupancy of NeRF and Point Cloud.
    $O_{NeRF} \leftarrow 1 - e^{-\delta \times \sigma}$
    $O_{NeRF} \leftarrow$ CLAMP($O_{NeRF}, min = 0, max = 1.1$)
    $L_{PCD} \leftarrow$ CE-LOSS(O_{NeRF}, X_{PCD}) $\times (1 - e^{-\frac{d}{2\sigma}})$
    return L_{PCD}
end function

B Segment Anything Hi3D

Our Hi3D is able to collaborate with other computer vision foundation models, such as Segment Anything (SAM) [37]. As indicated in Fig. 9, we first use SAM to extract the instance of our interest from the original image. Then we use an image captioning model, such as BLIP-2 [35], to get the corresponding text description of the instance image. The instance image is also used in Point-E

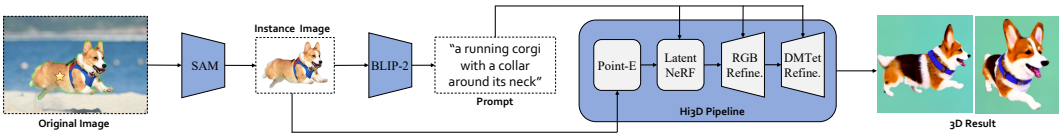


Figure 9: The SAM-Hi3D pipeline allows our Hi3D to work seamlessly with the foundational vision model, Segment Anything (SAM). This collaboration empowers users to create personalized 3D assets from common input images.

image-to-point-cloud model to generate the point cloud (See Section 3.2). The prompt is used as guidance to supervise the optimization of the entire Hi3D process. This sequence of steps is identified as the SAM-Hi3D pipeline. It has the capability to generate personalized 3D objects from an ordinary initial image.

C GPU cost conversion

As different text-3D frameworks utilize varying hardware to train their models, it creates a challenge to perform a fair comparison among the existing methodologies. For instance, DreamFusion [6] utilizes 4 TPUv4 cores, while Magic3D [9] employs 8 A100 GPUs, and Fantasia3D [25] makes use of 8 RTX3090 GPUs. In order to facilitate a fair comparison, we standardize all the costs to A100 GPU hours. Specifically, we equate 1 TPUv4 Hour to 1.4 A100 GPU Hours [4], and 1 RTX 3090 Hour to 0.5 A100 GPU Hours [5]. It is worth noting that the costs associated with parallel computing are overlooked in this comparison. Hence, we assume that one hour of training with 8 A100 GPUs is equivalent to eight hours of training with a single A100 GPU.

[4] https://cloud.google.com/blog/products/ai-machine-learning/cloud-tpu-v4-mlperf-2-0-results
[5] https://bizon-tech.com/gpu-benchmarks/NVIDIA-RTX-3090-vs-NVIDIA-A100-40-GB-(PCIe)/579vs592