Interactive3D: Create What You Want by Interactive 3D Generation

Shaocong Dong[1]*, Lihe Ding[2,4]*, Zhanpeng Huang[3], Zibin Wang[3],
Tianfan Xue[2]†, Dan Xu[1]†
[1]Hong Kong University of Science and Technology [2]The Chinese University of Hong Kong
[3]SenseTime Research [4]Shanghai AI Laboratory
{sdongae, danxu}@cse.ust.hk, {dl023, tfxue}@ie.cuhk.edu.hk
{wangzb02, yiyuanzhang.ai}@gmail.com, {huangzhanpeng}@sensetime.com

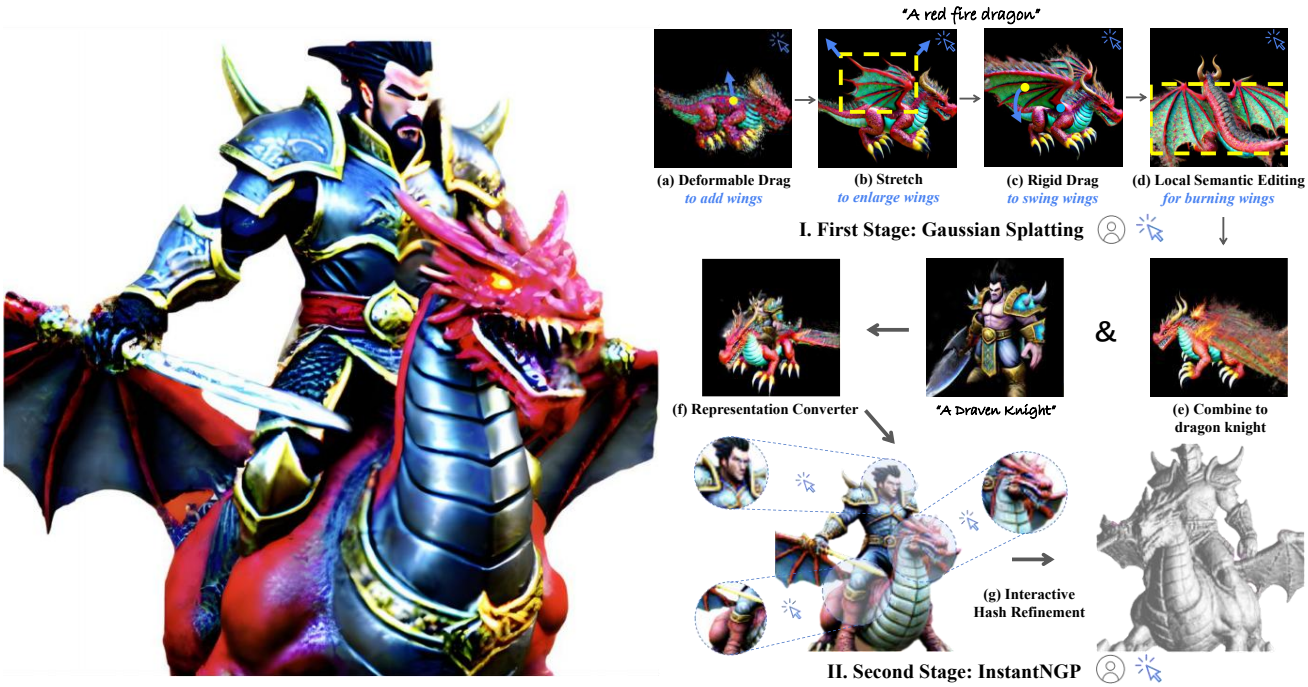


Figure 1. **Interactive 3D Generation**. **The first stage** involves using Gaussian ellipsoids for creating a base model where users can interact through different operations, such as deformable dragging to add features like wings, stretching to enlarge parts, rigid dragging to reposition elements, and local semantic editing to apply specific visual effects, e.g., making wings appear aflame. **The second stage** shows the conversion of this proposed Gaussian representation into an InstantNGP structure followed by an Interactive Hash Refinement process, allowing for further detailed enhancements. It demonstrates the framework's capability to merge and refine complex generations.

Abstract

*3D object generation has undergone significant advancements, yielding high-quality results. However, fall short of achieving precise user control, often yielding results that do not align with user expectations, thus limiting their applicability. User-envisioning 3D object generation faces significant challenges in realizing its concepts using current generative models due to limited interaction capabilities. Existing methods mainly offer two approaches: (i) interpreting textual instructions with constrained controllability, or (ii) reconstructing 3D objects from 2D images. Both of them limit customization to the confines of the 2D reference and potentially introduce undesirable artifacts during the 3D lifting process, restricting the scope for direct and versatile 3D modifications. In this work, we introduce **Interactive3D**, an innovative framework for interactive 3D generation that grants users precise control over the generative process through extensive 3D interaction capabilities. Interactive3D is constructed in two cascading stages, utilizing distinct 3D representations. The first stage employs Gaussian Splatting for direct user interaction, al-*

*Equal contribution.
†Corresponding author.

*lowing modifications and guidance of the generative direction at any intermediate step through **(i)** Adding and Removing components, **(ii)** Deformable and Rigid Dragging, **(iii)** Geometric Transformations, and **(iv)** Semantic Editing. Subsequently, the Gaussian splats are transformed into InstantNGP. We introduce a novel **(v)** Interactive Hash Refinement module to further add details and extract the geometry in the second stage. Our experiments demonstrate that proposed Interactive3D markedly improves the controllability and quality of 3D generation. Our project webpage is available at* https://interactive-3d.github.io/.

1. Introduction

Recent advancements [20, 29, 30] in 2D image generation, exemplified by approaches, such as diffusion models trained on extensive text-image paired datasets (e.g., LAION-series [31]), have made significant strides in aligning generated images with textual prompts. Despite this success, achieving precise control over image generation to meet complex user expectations remains a severe challenge. ControlNet [38] addresses this by modifying foundational 2D diffusion models with fine-tuning on specific conditional datasets, offering a subtle control mechanism guided by user-specific inputs.

On the other hand, 3D object generation, despite its promising progress [27, 35], confronts more intricate challenges than those encountered in 2D image generation. Although advancements have been observed from perspectives, including training 3D diffusion models on direct 3D datasets [10, 25], and lifting 2D diffusion priors to 3D representations (e.g., NeRF [21]) via techniques like SDS loss optimization [27], *precise control over the generated objects has not been fully achieved*. The reliance on initial text prompts or 2D reference images severely limits the generation controllability and often results in lower quality. The text prompts lack specificity to convey complex 3D designs accurately; while the 2D reference images can inform 3D reconstruction, they do not capture the full depth of 3D structures, potentially leading to various unexpected artifacts. Moreover, personalization based on 2D images lacks the flexibility that can be offered by direct 3D manipulation.

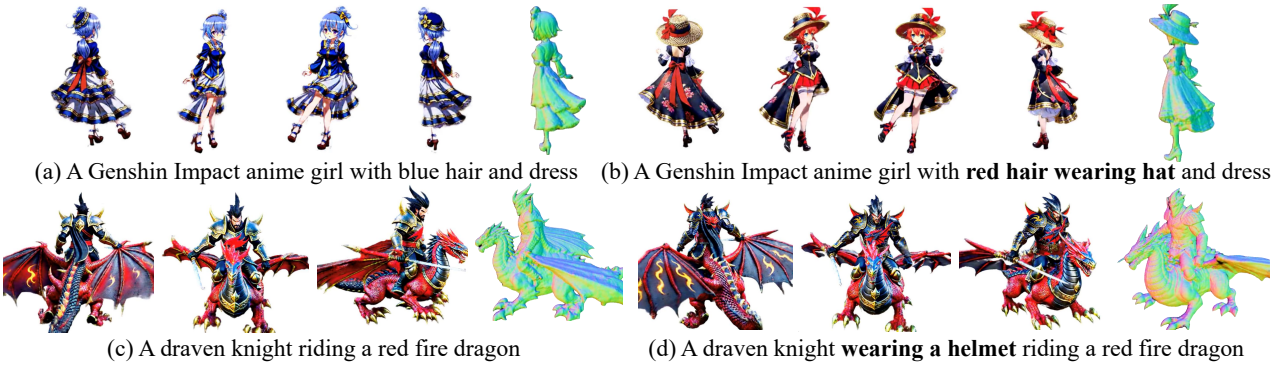
A straightforward idea to achieve controllable 3D generation is to adapt the ControlNet to 3D generation. However, this strategy encounters significant obstacles: (i) the control signals for 3D are inherently more complex, making the collection of a conditioned 3D dataset exceptionally challenging when compared to the 2D paradigm; (ii) the absence of powerful foundational models in the 3D domain, like stable diffusion for 2D [20], impedes the possibility of developing fine-tuning techniques at this time. These hurdles suggest the need for a different strategy. As a result, we are inclined to explore a novel question: *can we directly integrate flexible human instructions into the 3D generation?*

To address the above-identified challenges, we introduce *Interactive3D*, a framework devised to facilitate user interaction with intermediate outputs of the generative process, enabling precise control over the generation and effective enhancement of generated 3D object quality. Our approach is characterized by a two-stage process leveraging distinct 3D representations. Specifically, the first stage utilizes the SDS loss [27] to optimize a 3D object by Gaussian Splatting [11] representation, which allows for independent user modifications during the optimization process. The second stage transforms the Gaussian representation into InstantNGP [23] structures and applies proposed Interactive Hash Refinement for detailed textures and 3D geometry.

For the first stage, the Gaussian blobs enable direct user feedback, such as the addition and the removal of object components by manipulating the Gaussian blobs, as illustrated in the knight and dragon composition in Fig. 1, and a dragging operation based on the principles of DragGAN [26]. Users can select a source and a target point within the 3D space, which guide the Gaussian blobs (conceptualized as point clouds) from an original to a desired location using a *motion-supervision loss*. Notably, our framework facilitates both deformable and rigid dragging of object elements through a *rigid constraint loss*, allowing users to create new or adjust existing 3D model components.

Furthermore, our *Interactive3D* allows for precise selection of subsets of Gaussian blobs, to which users can apply various transformations, e.g., the stretch operation shown in Fig. 1 (b). By restricting the gradient flow to these subsets, we enable focused optimization of chosen 3D parts using modified text prompts, without altering other areas of the model. This strategy is termed as local semantic editing. Its capability is exemplified in Fig. 1 (d), where the wings are edited to simulate the effect of being ablaze. To enhance the guidance from user interactions during generation, we integrate an *interactive SDS loss* with an adaptive camera zoom-in technique, significantly improving the optimization efficiency of the modified 3D areas.

At the beginning of the second stage, the modeled Gaussian representations are transformed into InstantNGP structures through a swift NeRF distillation technique. This strategic transformation synergies the strengths of both representations: Gaussian blobs, while more friendly for direct editing, face challenges in reconstructing high-quality 3D geometry; on the other hand, InstantNGP structures excel in providing a foundation for further geometry refinement and mesh extraction, as depicted in Fig. 1 (f). Given that the converted InstantNGP employs hash tables to associate learnable features with 3D grids, we have developed an innovative *Hash Refinement Module*. This module enables the interactive enhancement and detailing of chosen areas



(a) A Genshin Impact anime girl with blue hair and dress (b) A Genshin Impact anime girl with **red hair wearing hat** and dress

(c) A draven knight riding a red fire dragon (d) A draven knight **wearing a helmet** riding a red fire dragon

Figure 2. Qualitative generation results of the proposed Interactive3D. We achieve high-quality and controllable 3D generation.

within the base radiance fields, as demonstrated in Fig. 1 (g). Specifically, we begin by extracting coarse occupancy grids from the initial radiance field. When a user selects an area for optimization, a candidate grid set is determined by the intersection of this area with the occupancy grids. These grids are then categorized into multi-level sets with finer resolutions than the original field, and multiple refinement hash tables are constructed to map the selected area to learnable refinement features, which are then fed into lightweight MLPs to capture residual colors and densities, concentrating detail enhancement on local surface regions.

Through our Interactive Hash Refinement module, users can precisely control the refinement process by selecting specific areas for optimization and adjusting the levels of representation details, such as the hash table levels and the capacities. Hence, users can sculpt their envisioned objects during generation by employing the array of interactive options provided by *Interactive3D*.

In summary, our contributions are threefold:

- **Framework Deisgn.** We introduce a novel interactive 3D generation framework, *Interactive3D*, which empowers users to precisely control the 3D generation process via direct instructions.
- **3D Representations.** We reveal that incorporating the Gaussian Splatting for user interaction and the Instant-NGP for further geometry refinement and mesh extraction leads to higher-quality 3D generation.
- **Generation Results.** With precise control and effective user feedback in the generation process, our method significantly improves the generation quality.

2. Related Work

2.1. 3D Generative Models

3D Generation is a foundation task and has been studied widely. Early works focus on the 3D representations including voxels [5, 7, 17, 33, 36], point clouds [1, 22, 37], meshes [6, 32, 39] and implicit fields [3, 19]. Recently, diffusion models [8] have achieved great success in 2D content creation [24, 28–30], and there has been substantial research into 3D diffusion, which has significantly improved 3D generation. These 3D diffusion models can be divided into two directions: optimization-based [27] and feed-forward methods [10]. For the optimization-based methods, the DreamFusion series [9, 13, 20, 27, 34] design an SDS loss based on probability density distillation, which enables the use of a 2D diffusion model as a prior for optimization of a parametric image generator (e.g., rendered from NeRF [21]). They optimize a 3D consistent radiance field via gradient descent such that its 2D renderings from random angles achieve low losses. These optimization-based methods can realize zero-shot and high-quality generation with several hours of optimization. Further works, such as Prolificdreamer [35] and Fantasia3d [2], achieve higher quality in 3D content generation by modifying the SDS loss. For feed-forward methods, PointE [25] generates a single synthetic view using a text-to-image diffusion model, and then produces a 3D point cloud using a second diffusion model that is conditioned on the generated image. The follow-up work Shap-E [10] trains a latent diffusion model on NeRF's parameters. One-2-3-45 [14] uses Zero-1-to-3 [15] to generate multi-view images, which are fed into a SparseNeuS [16] for 3D generation. However, current methods [2, 9, 13, 20, 27, 34, 35] only rely on initial text prompts or reference images to 3D generation, restricting their controllability. In contrast, our framework introduces user interactions into the optimization, achieving flexible and controllable 3D generation.

2.2. Gaussian Splatting

The recent 3D representation Gaussian Splatting [11] has revolutionized the 3D rendering field by using a set of Gaussian blobs to represent the 3D scenes and splat them onto the image plane to obtain renderings. Compared with Neural Radiance Field (NeRF), Gaussian splatting models the 3D world by explicit Gaussian blobs, which are flexible and naturally suitable for human interactions and editing.

2.3. InstantNGP

To improve the quality and efficiency of NeRF, [23] proposes InstantNGP which utilizes multi-level hash tables to

map between learnable features and 3D query positions. InstantNGP does not handle the hash conflict explicitly and makes the gradients to guide the optimization of features. In this case, the surface positions obtain larger gradients during training and dominate the feature updates. Thanks to the adaptive feature update by hash mapping, InstantNGP can represent the 3D world by more fine-grained grids and easily extract 3D geometries (e.g., mesh) from the continuous radiance field. However, InstantNGP is still an implicit representation and is hard to interact with or edit.

3. The Proposed Interactive3D Approach

Achieving Interactive 3D Generation is non-trivial and poses two core challenges: (i) we need to design an efficient and effective interactive mechanism for the 3D representation to control the generation process, and (ii) we need to obtain high-quality 3D outputs (e.g., the mesh) aligned with users' expectations. As for the 3D representation, NeRF [21] can output high-quality 3D objects, however, its implicit representation makes it hard to incorporate interactions. While the recent 3D Gaussian Splatting is naturally suitable for interaction due to the independence and explicitly of Gaussian blobs, it cannot output the commonly used mesh with high quality so far. Observing the complementarity of these two representations, we design a two-stage interactive 3D generation framework, where Stage I (Sec. 3.1) adopts Gaussian Splatting to achieve flexible interactions and Stage II (Sec. 3.2) converts the Gaussian blobs to InstantNGP for further geometry refinement and mesh extraction, as shown in Fig. 3. Below we first introduce the first interactive generation stage with Gaussian Splatting.

3.1. Interaction with Gaussian Splatting

In the first stage, we represent a 3D object as a set of N Gaussian blobs $\mathcal{E} = \{(c_i, o_i, \mu_i, \Sigma_i)\}_i^N$, where $c_i, o_i, \mu_i, \Sigma_i$ represents the color, opacity, position, and the covariance of the i-th Gaussian, respectively. To incorporate interactions, we treat each Gaussian blob as a point and formulate a point cloud set $\mathcal{S} = \{\mu_i\}_i^N$. Thanks to the flexibility of treating Gaussian blobs as points, we can introduce various user interactions by explicitly modifying the point set $\mathcal{S}$, as discussed in Sec. 3.1.1, Sec. 3.1.3, and Sec. 3.1.4. Once the adjustment is finished, we adopt an interactive SDS loss to optimize the modified parts efficiently in Sec. 3.1.5.

3.1.1 Adding and Removing Parts

To add parts, suppose we have two Gaussian blob sets $\mathcal{E}_1$ and $\mathcal{E}_2$, we can simply obtain the combined set by concatenating them. In practice, this interaction often happens when the user desires to combine two objects (e.g., knight and dragon as shown in Fig. 1).

To remove parts P, we can directly delete the Gaussian blobs within the part by:

$$\mathcal{E}' = \mathcal{E} - \{e_i\}_{i \in I(P)}\,, \tag{1}$$

where $I(p) = \{i : \mu_i \in P\}$ indicates the indexes of Gaussian blobs in P. Importantly, the determination of whether blob e_i belongs to P depends on how we define the parts. In practice, we offer two ways to select a part. The first one is defined in 2D images. Specifically, the users can choose a point on the rendered 2D images from Gaussian Splatting and utilize an off-the-shell segmentor (e.g., SAM [12]) to obtain multi-view part masks. Subsequently, we project all the points in $\mathcal{S}$ onto all masked views and obtain the blobs with all projections within masks as the part blobs. We use 2 views by default. However, such a 2D mask-based part selection needs to infer with a large-scale pretrained segmentation network, which may be accurate but not efficient. Furthermore, if we do not need a precise part set or we select parts at the early stage of generation (the rendered images may not be informative enough for SAM), we can directly select points in 3D.

3.1.2 Geometry Transformation

Given that any interested part can be selected as discussed in Sec. 3.1.1, we can take a further step to construct a bounding box B for each P. In this way, all the traditional geometry transformations $\mathcal{T}$ such as rotation, translation, and stretching can be first applied to the part P and then it is concatenated with the unchanged set as follows:

$$\mathcal{E}_{\text{trans}} = \mathcal{T}(B(P)) + \mathcal{E}'. \tag{2}$$

3.1.3 Deformable and Rigid Dragging

Although we have achieved the part-level geometry transformations, the users may prefer more flexible and direct interactions. Inspired by DragGAN [26], we propose deformable and rigid 3D Dragging operations. The deformable dragging regards the local structure as a plasticine and aims to deform the geometry smoothly, e.g., dragging new wings from the dragon's back, while the rigid dragging treats the local region as a rigid part and forces the local structure unchanged. Specifically, we first select a source point p_s, a target point p_t, and a local region radius r. Then, the activated local object part can be written as:

$$P = \{e_i | \|\mu_i - p_s\|_2 \leq r\}\,. \tag{3}$$

Subsequently, we move all points with a minor offset along the direction from p_s to p_t before each optimization step:

$$\mu_i' = \mu_i + \alpha \frac{(p_t - p_s)}{\|p_t - p_s\|_2}, \tag{4}$$

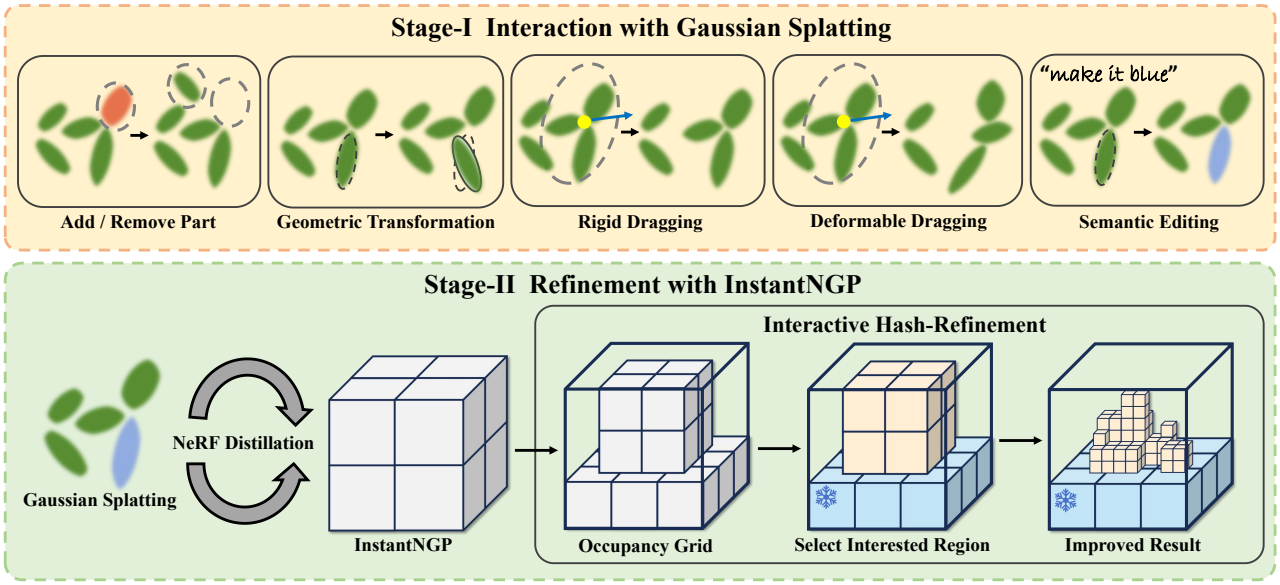


Figure 3. **The overall architecture of Interactive3D.** It contains two stages with distinct 3D representations: **(I)** Gaussian Splatting for flexible user interactions such as add/remove parts, geometry transformation, deformable or rigid dragging and semantic editing; **(II)** the Gaussian blobs are converted to InstantNGP using NeRF distillation and fine-tuned by our Interactive Hash Refinement Module.

where α is a predefined hyper-parameter to control the movement step. Then the modified blobs $\{e_i' = (c_i, o_i, \mu_i', \Sigma_i)\}_{i=1}^{N}$ can render images through Gaussian Splatting. From the rendered images, we can compute the SDS loss L_{SDS} following [4, 27]. Meanwhile, to encourage the part P to move towards p_t during optimization, we add a motion supervision loss following [26]:

$$L_{\text{motion}} = \sum_{i \in I(P)} (\|\mu_i' - p_t\|_1), \tag{5}$$

where $I(P)$ indicates a set of indexes of Gaussian blobs in P. Then we can compute the gradient and update the parameters of each Gaussian blob. Note that we do not need a point tracking process as in [26] since the 3D points are naturally tracked. Importantly, we apply densification and pruning operations as in [11] to fill the gaps and eliminate noises after the dragging operation.

To achieve deformable or rigid dragging, we made two adjustments to the above process. First, for the deformable dragging, we continuously update the activated part P by updating the current position of source point p_s as in Eq. (3), which allows the newly generated points to be dragged, leading to structure deformation. For the rigid dragging, the part P is fixed to encourage an integral movement. Second, we introduce a rigid constraint loss to enforce the rigid moving without the deformation:

$$L_{\text{rigid}} = \sum_{i \in I(P)} |\|p_s - p_i\|_2 - \|p_s^* - p_i^*\|_2|, \tag{6}$$

where p_s^* and p_i^* are the initial position of the source point and its neighboring points, respectively. L_{rigid} encourages the distance between the neighboring points and the source point to remain unchanged, leading to integral movement.

In general, the user can flexibly combine the deformable and rigid dragging operations by switching the above adjustments and accordingly modifying the hyperparameters (e.g., the movement step α and the loss weight for L_{rigid}).

3.1.4 Semantic Editing

Sometimes the users may desire to interact with the generation process by simple text prompts. To achieve this, we further propose a semantic editing operation. Specifically, we can first select a part, and then input a new text prompt (e.g., make the wings burning) to compute the SDS loss, and optimize the parts correspondingly to match the newly added semantic features. This operation can be better accomplished with an Interactive SDS loss discussed in the following section.

3.1.5 Interactive SDS Loss

Once the above modifications are finished, we propose to utilize an interactive SDS loss to efficiently optimize the Gaussian blobs. First, observing that the user may use part-level interactions in most cases, we do not need to render the whole view of the object. Instead, we adopt an adaptive camera zoom-in strategy by putting the camera near the modified region and computing the SDS loss using lo-

cal renderings and user-modified text prompts (Sec. 3.1.4), resulting in significantly improved optimization efficiency. Furthermore, since we can interact with the entire generation process, we make the denoising step t in the SDS loss adjustable for the user to adapt to different stages. For example, in the early stage with noisier shapes, t can be set to a large number (e.g., 0.98), while when we aim to fine-tune some parts, t can be set to a small number (e.g., 0.3) to avoid drastic changes.

3.2. Refinement with InstantNGP

Once obtaining the generated Gaussian blobs $\mathcal{E}$ in Stage-I, we can first convert it to the InstantNGP [23] representation by a NeRF distillation operation Sec. 3.2.1 and then use the Hash Refinement module Sec. 3.2.2 for further fine-tuning.

3.2.1 NeRF Distillation

Different from the reconstruction task, we find that the Gaussian blobs cannot generate fine-grained 3D objects under the supervision of the SDS loss, often with long trip noises and artifacts. We attribute this to the unstable optimization process for independent Gaussian blobs when supervised by unstable SDS losses. In addition, extracting meshes or other geometries from Gaussian Splatting is still unsolved. Therefore, to further improve the quality and extract geometry, we convert the Gaussian blobs into the InstantNGP representation. Specifically, we adopt a simple yet effective distillation approach by supervising random renderings from the InstantNGP with the images rendered by Gaussian Splatting:

$$L_{\text{distill}} = \frac{1}{M} \sum_{c \sim C(\theta,\phi)} \|\mathcal{V}(\mathcal{F}, c) - \mathcal{R}(\mathcal{E}, c)\|_1, \quad (7)$$

where c is the camera pose derived from a predefined pose distribution $C(\theta, \phi)$; $\mathcal{R}$ is the Gaussian Splatting rendering process; $\mathcal{V}$ and $\mathcal{F}$ are the volumetric rendering and parameters of InstantNGP, respectively.

3.2.2 Interactive Hash Refinement

Upon obtaining the converted InstantNGP representation $\mathcal{I} = \{\mathcal{F}, \mathcal{H}\}$, where $\mathcal{H} = \{H_k\}_{k=1}^{L}$ contains L-level hash tables and $\mathcal{F} = \{F_k\}_{k=1}^{L}$ contains corresponding L-level features, we aim to interactively refine $\mathcal{I}$ to further improve the generation quality. To achieve this, a simple way is to select the unsatisfying regions and put the camera near these areas for further refinement like what we did in Sec. 3.1.4. However, unlike Gaussian Splatting, which can grow new blobs with infinite resolutions and numbers, the informative features stored in InstantNGP have limited capacity (e.g., finite grid resolution and hash table length), leading to refinement bottleneck. Furthermore, InstantNGP does not handle hash conflicts so that different parts may enjoy the same features, resulting in the degeneration of other parts when fine-tuning one part.

To handle these problems, we propose an Interactive Hash Refinement module, which fixes the original InstantNGP and adds new learnable residual features to the interested regions, overcoming the hash conflict problem by introducing part-specific hash mapping. The Interactive Hash-Refinement strategy (i) adaptively adds informative features to different regions. For example, we can add hash tables with more levels and features for highly complex regions, while adding fewer residuals in regions that only need to remove artifacts, and the strategy can also (ii) make the model focus on worse regions while freezing the satisfying parts without introducing degeneration.

Specifically, we first extract the binary occupancy grids O with a resolution of $32 \times 32 \times 32$ from the converted $\mathcal{I}$. Given the interested region Q defined by a center o and radius r:

$$Q = \left\{p \middle| \|p - o\|_2 \leq r, \forall p \in \mathbb{R}^3\right\}. \quad (8)$$

We intersect Q and O to obtain a part occupancy region $O^{\text{part}} = O \cap Q$. Then we divide O^{part} to multi-resolution grids and establish a part-specific multi-level hash table set $\mathcal{H}^{\text{part}}$ and learnable features $\mathcal{F}^{\text{part}}$, mapping solely from the local region to local features, avoiding sharing information with other object parts:

$$f_k = \begin{cases} H_k^{\text{part}}(p, F_k^{\text{part}}), & p \in O^{\text{part}} \\ 0, & p \notin O^{\text{part}} \end{cases}, \quad (9)$$

where f_k denotes the mapped features at level k at position p. Subsequently, we introduce new lightweight MLPs to convert newly added features to residual densities and colors to influence the final renderings. After that, we can use the interactive SDS loss to optimize the local regions as discussed in Sec. 3.1.5. .

4. Experiment

4.1. Implementation Details

We build *Interactive3D* in two stages with different 3D representations. We follow [4] to implement the Gaussian Splatting-based optimization in Stage I and follow [23] to implement InstantNGP-based refinement in Stage II. We implement the Hash Refinement module in parallel with CUDA to improve efficiency. The general training step for one object is 20k (10k for Stage I and 10k for Stage II). We conduct our experiments on NVIDIA A100 GPUs. More details can be found in the supplementary materials.

4.2. Qualitative Results

We show some 3D generation results from *Interactive3D* as well as the interaction process in Fig. 6, Fig. 11, and Fig. 4

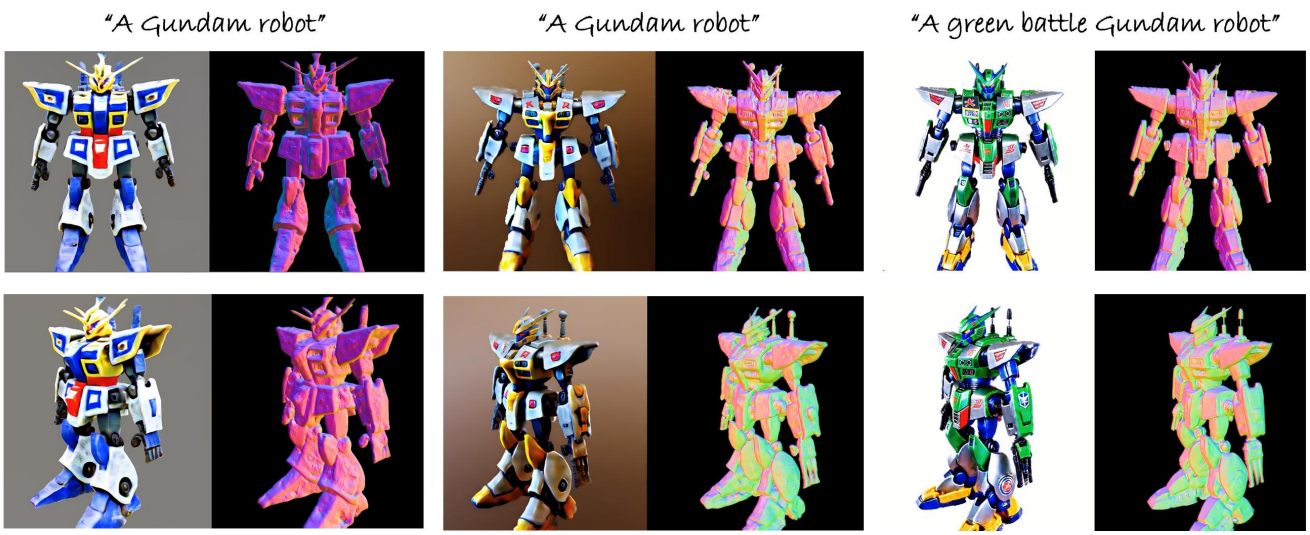


Figure 4. The effectiveness of Interactive Hash Refinement. (a) Results with the original hash table length and resolution from InstantNGP. (b) Results with more refined hash tables and features while without part-specific hash mapping. (c) Results with complete Interactive Hash Refinement module.

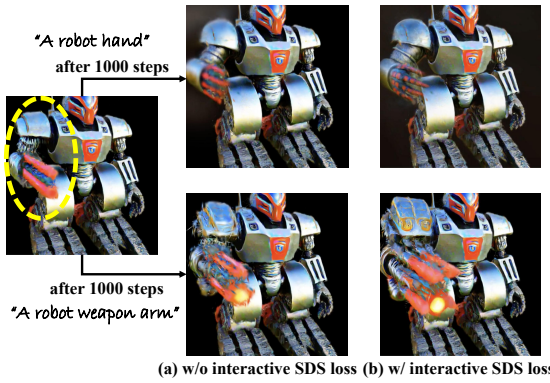


Figure 5. Ablation study of the proposed Interactive SDS loss.

(c). As shown in the first row of Fig. 6, we can interrupt the generation process and change the pose of the generated human, and then continue to optimize. It can be observed that the standing 3D man is smoothly converted to a kick-dancing man by directly modifying the Gaussian blobs. The dancing man is then optimized by a few steps to fix some artifacts and holes after the interaction, and it is converted to InstantNGP for further refinement. Compared with the results directly generated from current state-of-the-art methods, our generated objects are more controllable with better geometry and texture. For instance, ProlificDreamer has totally wrong geometry and mismatches the text prompt. In the second row of Fig. 6, we rigidly drag the head of Tyrannosaurus Rex from looking forward to the right direction, achieving a DragGAN [26]-style interaction while in 3D. In Fig. 4 (c), we show that by applying our Interactive Hash Refinement module to a coarse 3D object, *a Gundam robot* in this case, we can achieve significant textural and geometric improvements. It is noteworthy that the interactions can be combined in one generation process as shown in Fig. 1. More results can be found in the supplementary material.

Table 1. Quantitative comparison on the CLIP R-Precision [10].

Method	R-Precision	Average Time
DreamFusion [27]	0.67	**1.1h**
ProlificDreamer [35]	0.83	3.4h
Ours	**0.94**	**50min**

4.3. Quantitative Results

Following [10], we use the CLIP R-Precision to quantitatively evaluate our generated results in Tab. 1. We use 50 prompts derived from Cap3D [18] and compare the results with other methods. *Interactive3D* achieves the highest CLIP R-Precision by interactive generation, which demonstrates the strong controllability of our method. Meanwhile, we can achieve highly efficient 3D generation, because of the following two reasons: (i) we adopt the fast Gaussian Splatting in Stage I and utilize its results to initialize the optimization in Stage II which speeds up the convergence; (ii) the interactions are incorporated into some checkpoint of the original optimization process without the need for extra training steps, we achieve efficient 3D generation.

4.4. Ablation Studies

Now, we investigate the impact of the Rigid Constraint loss, the Interactive SDS loss, and the Hash Refinement Module. More ablations can be found in the supplementary material.

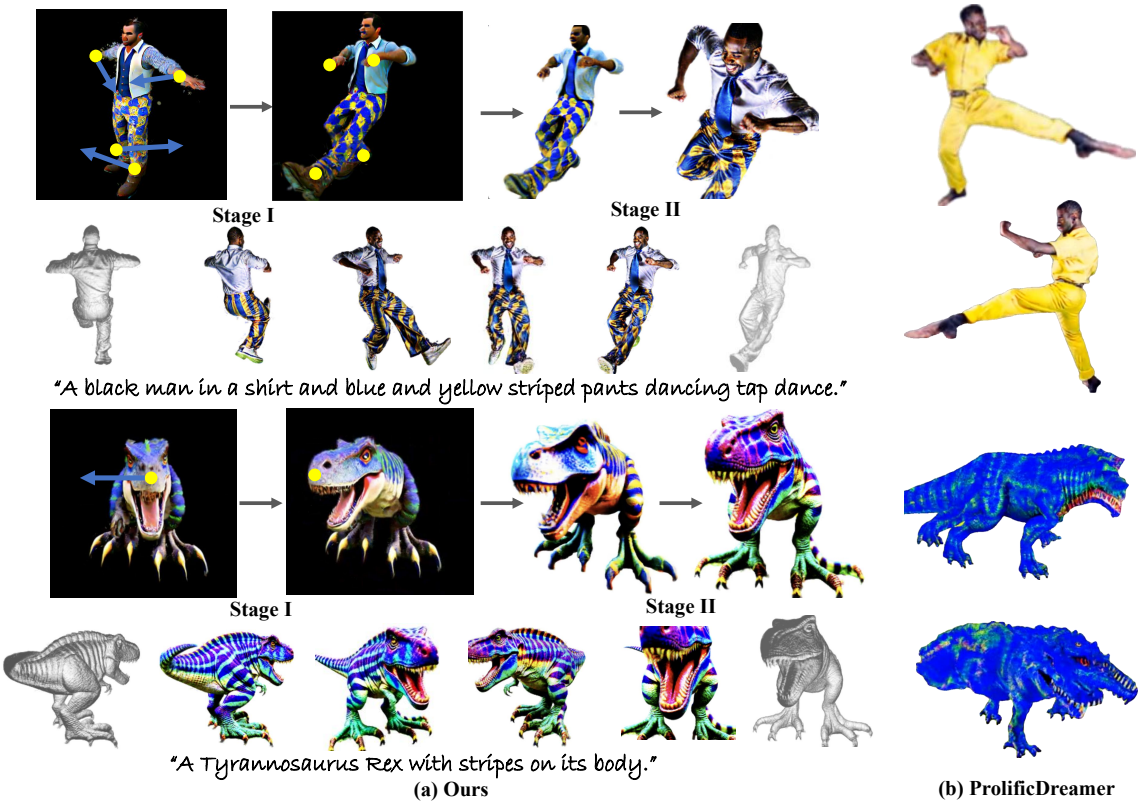


Figure 6. Qualitative results of the rigid dragging operation, demonstrating the effectiveness and controllability of user interactions.

(a) original (b) w / o rigid constraint (c) w / rigid constraint

Figure 7. Ablation study of the proposed Rigid Constraint loss.

Effect of Rigid Constraint Loss As shown in Fig. 7, the Rigid Constraint loss plays a vital role in ensuring the local structure unchanged during the dragging. With this constraint, the local structure can be well maintained during the dragging while eliminating this loss leads to deformation.

Effect of Interactive SDS Loss We ablate the Interactive SDS loss in Fig. 5. Given a robot in the representation of Gaussian blobs, we aim to change its original arm to other formats. The results show that we achieve better-modified results by using the Interactive SDS loss after the same number of training steps compared with the baseline.

Effect of Interactive Hash Refinement Module We ablate the Interactive Hash Refinement Module and the effectiveness of part-specific hash mapping in Fig. 4. Refer to Fig. 4 (a), it shows that the generation quality is blurry and lacks details with only the original SDS optimization, and the details and artifacts cannot be improved even with extremely long optimization steps (e.g., 100k training iterations). The integration of the proposed interactive refinement enhances the realism of models by correcting unreasonable components, such as the robot head in Fig. 4 (b). The proposed part-specific hash mapping further advances the texture and geometric precision of the model dramatically, as shown in Fig. 4 (c). Furthermore, the part-specific hash mapping is also important to avoid region conflicts, leading to more fine-grained generation results.

5. Conclusion

In this paper, we proposed *Interactive3D*, a novel framework for controllable and high-quality 3D generation. Interactive3D works in two main stages, using different 3D representations. In Stage I, we use Gaussian Splatting representation which allows users to flexibly change the model by adding or removing parts, transforming shapes, and making semantic editing. In Stage II, we first convert the Gaussian blobs to InstantNGP by fast NeRF distillation, and then propose a novel Interactive Hash Refinement to further improve the quality and extract 3D geometry.

Acknowledgments: This work was supported in part by Research Grants Council (RGC) of the Hong Kong SAR under grant No. 26202321, HKUST startup fund No. R9253, and CUHK Direct Grants (RCFUS) No. 4055189. We also gratefully acknowledge the support of SenseTime.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 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] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 3

[4] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585*, 2023. 5, 6, 2

[5] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *3DV*, 2017. 3

[6] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *NeurIPS*, 2022. 3

[7] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. In *ICCV*, 2019. 3

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 3

[9] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *CVPR*, 2022. 3

[10] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 2, 3, 7

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 2, 3, 5

[12] 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. In *ICCV*, 2023. 4

[13] 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

[14] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *NeurIPS*, 2024. 3

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

[16] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *ECCV*, 2022. 3

[17] Sebastian Lunz, Yingzhen Li, Andrew Fitzgibbon, and Nate Kushman. Inverse graphics gan: Learning to generate 3d shapes from unstructured 2d data. *arXiv preprint arXiv:2002.12674*, 2020. 3

[18] Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. Scalable 3d captioning with pretrained models. *NeurIPS*, 2024. 7

[19] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 3

[20] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *CVPR*, 2023. 2, 3

[21] 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. 2, 3, 4

[22] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019. 3

[23] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41 (4):102:1–102:15, 2022. 2, 3, 6

[24] 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. 3

[25] 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. 2, 3

[26] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 2, 4, 5, 7

[27] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 2, 3, 5, 7

[28] 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. 3

[29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2

[30] 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. *NeurIPS*, 2022. 2, 3

[31] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021. 2

[32] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *NeurIPS*, 2021. 3

[33] Edward J Smith and David Meger. Improved adversarial systems for 3d object generation and reconstruction. In *CoRL*, 2017. 3

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

[35] 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. *NeurIPS*, 2024. 2, 3, 7

[36] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *NeurIPS*, 2016. 3

[37] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, 2019. 3

[38] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 2

[39] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. *arXiv preprint arXiv:2010.09125*, 2020. 3