Vinedresser3D: Towards Agentic Text-guided 3D Editing

Yankuan Chi[1*] Xiang Li[2*] Zixuan Huang[2] James M. Rehg[2]

[1]The Hong Kong University of Science and Technology
[2]University of Illinois Urbana-Champaign

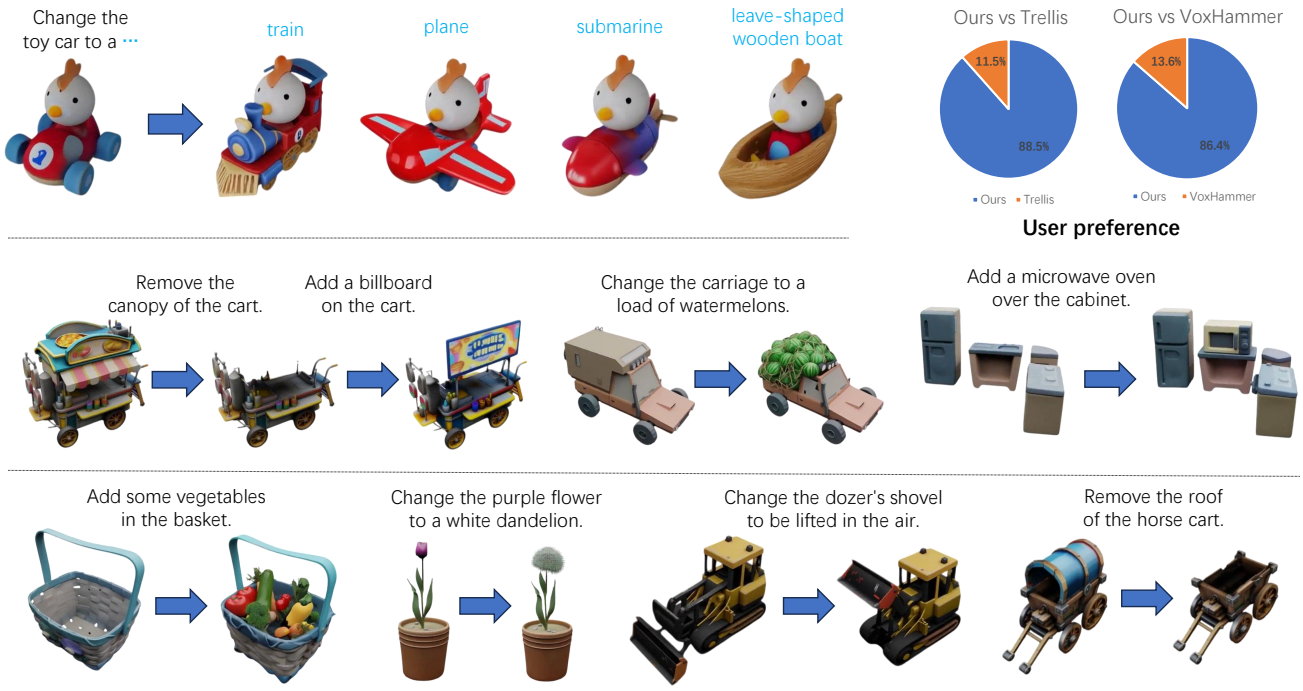

Figure 1. We propose Vinedresser3D , an agent that can intelligently perform high-quality text-guided 3D editing. It can handle various kinds of edits (addition, modification and deletion), support multi-turn editing and tackle different types of 3D assets (objects and scenes).

Abstract

Text-guided 3D editing aims to modify existing 3D assets using natural-language instructions. Current methods struggle to jointly understand complex prompts, automatically localize edits in 3D, and preserve unedited content. We introduce Vinedresser3D, an agentic framework for high-quality text-guided 3D editing that operates directly in the latent space of a native 3D generative model. Given a 3D asset and an editing prompt, Vinedresser3D uses a multimodal large language model to infer rich descriptions of the original asset, identify the edit region and edit type (addition, modification, deletion), and generate decomposed structural and appearance-level text guidance. The agent then selects an informative view and applies an image editing model to obtain visual guidance. Finally, an inversion-based rectified-flow inpainting pipeline with an interleaved sampling module performs editing in the 3D latent space, enforcing prompt alignment while maintaining 3D coherence and unedited regions. Experiments on diverse 3D edits demonstrate that Vinedresser3D outperforms prior baselines in both automatic metrics and human preference studies, while enabling precise, coherent, and mask-free 3D editing.

1. Introduction

Editing 3D assets is a fundamental problem in 3D computer vision, with broad applications in digital content creation, virtual and augmented reality, and robotics. While recent

text-guided 3D generation methods greatly lower the barrier to creating 3D content from scratch, in practice, high-quality 3D editing still relies heavily on professional artists and manual tools. This process is labor-intensive, requires domain expertise, and scales poorly with the growing demand for customized 3D content. These limitations motivate automatic systems that can follow high-level textual user instructions and perform precise 3D edits in an intelligent and reliable way.

Despite this great progress in 3D editing research, current text-guided 3D editing methods still struggle to semantically understand complex editing requests, automatically detect precise 3D editing regions from textual instructions alone, and perform high-quality 3D editing that both follows prompts closely and preserves unedited regions. Existing text-guided 3D editing methods can be broadly grouped into three lines, each with important limitations. Score Distillation Sampling-based approaches [9–12, 29, 41, 47, 52, 68] optimize a 3D representation under the guidance of 2D diffusion models, but per-scene optimization is computationally expensive and typically requires careful tuning to avoid unintended global changes. A second line follows a "2D editing + 3D reconstruction" paradigm [5, 7, 14, 17, 26, 43, 54, 56]. These works first edit rendered views with multi-view diffusion and image editing models and then reconstructing the edited asset with 3D reconstruction models. These methods are fundamentally constrained by multi-view inconsistency and incomplete 3D supervision, often struggling with the quality of unobserved regions and preservation of unedited geometry. Concurrent to our work, VoxHammer [28] builds on a native 3D generative model [59] to perform editing directly in 3D, but still requiring user-provided 3D masks, and cannot follow complex edit requests.

Recent advances in multimodal large language models (MLLMs) [2], image editing models [1], 3D segmentation [31], and strong 3D flow-based native generative models [59] have significantly improved our ability to understand text instructions, manipulate images, and operate directly in 3D latent space rather than through 2D proxies. Building on these developments, we argue that a natural next step is to move from *single-model* methods to a *3D editing agent* that integrates multiple specialized tools. Such an agent should (1) understand high-level text instructions, (2) automatically localize the intended editing region in 3D, and (3) invoke appropriate tools to carry out the edit while preserving the geometry and appearance of the rest of the asset.

We propose Vinedresser3D, an agent for high-quality text-guided 3D editing (Figure 1). Vinedresser3D is built around a core of MLLM (Gemini-2.5-flash [2]). It takes as input a 3D asset and a textual editing prompt. It first uses the MLLM to parse the instruction, reason about the desired changes, and produce detailed textual guidance for the edit. Then our agent selects a representative view and invokes an image editing model to obtain an edited image as visual guidance for 3D editing. To ensure the 3D editing does not make unintended local changes, our agent employs a 3D grounding pipeline that automatically detects the editing region in the 3D asset based on the text prompt, eliminating the need for user-provided 3D masks. Finally, leveraging the latent space of native 3D generation models such as Trellis [59], Vinedresser3D performs precise inversion–based editing in 3D latent space, enabling faithful editing prompt alignment while preserving the geometry and appearance of unedited regions, producing a high-quality and holistic 3D edit.

In summary, our main contributions are:

- We introduce Vinedresser3D, a 3D editing agent that uses an MLLM as its core to intelligently interpret text instructions and coordinate a set of tools for high-quality 3D editing. Our framework achieves strong text alignment, robust preservation of unedited parts, and high overall 3D quality.
- We demonstrate that an MLLM trained primarily on 2D image–text data can be integrated into a 3D editing pipeline, where it plans editing strategies, generates multi-modal guidance, and interacts with 3D segmentation, image editing, and 3D generation tools, to greatly improve the editing performance.
- We conduct extensive experiments on text-guided 3D editing, showing that Vinedresser3D provides precise, coherent edits and compares favorably against state-of-the-art 3D editing baselines in both quantitative metrics and human evaluations.

2. Related Work

LLM Agents. LLM Agents are of great research interest in the AI community in recent years [4, 36, 48, 49, 57, 58, 63], since having an agent to automatically and smartly handle users' requests would be very convenient and efficient. Moreover, agents can utilize the strong knowledge learned by large AI models to solve users' problems accurately and effectively. In 3D generation and editing fields, many previous works [6, 8, 16, 18, 22–24, 35, 40, 50, 62, 65, 66] have tried to develop intelligent agents to tackle the 3D tasks. However, no powerful agent for 3D editing with a text prompt as the pure input exists yet. So we believe developing an agent to intelligently perform text-guided 3D editing is of great importance.

Inversion-based editing. Due to the limited amount of paired training data of editing, many editing pipelines [19, 27, 33, 37–39, 45, 51, 53] adopt the inversion-editing methodology. It inverses the original object into the initial structured noise using a diffusion or flow model. It then edits the object following the generation process with new

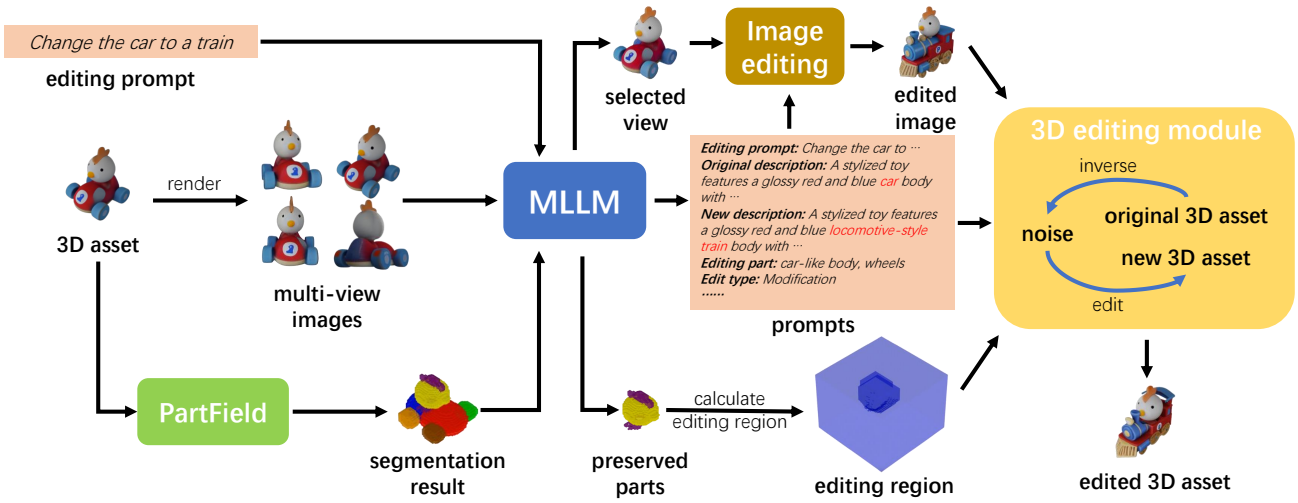

Figure 2. Pipeline overview. Given a 3D asset and an editing prompt, Vinedresser3D uses an MLLM to obtain new text and image guidance, automatically detects the intended editing region and then performs precise editing through an inversion-editing module.

conditions towards the editing target. The editing process can either inject the features or attention maps of the inversion trajectory [19, 51, 53] or inpaint the object [34]. We choose to use inpainting as we believe it can provide a greater control over parts we want to alter or preserve. Moreover, most inversion-based methods are for 2D editing because there are few powerful 3D diffusion or flow models. However, we want to directly operate in the 3D space, so we choose a start-of-the-art flow-based 3D generation model [59] as our baseline.

3D editing. Due to the challenges of manipulating the 3D space and the lack of 3D data, most existing 3D editing methods actually heavily rely on the 2D domain. One stream of methods [9–12, 29, 41, 47, 52, 68] utilizing Score Distillation Sampling [42] to update the 3D asset. They repeatedly render images and propagate the gradients from a 2D diffusion model back to the 3D asset to optimize it. However, this process is time-consuming and computationally expensive. Another stream of methods [5, 7, 14, 17, 26, 43, 54, 56] render multi-view images of the original assets and edit the images with multi-view diffusion models. They then reconstruct the edited asset from those images using reconstruction techniques. This stream of methods usually suffers from multi-view inconsistency and the lose of spatial information due to occlusion and distortion of rendering. Considering the problems with relying on the 2D domain, some methods [3, 46, 60, 64] propose to encode the 3D asset into a latent space and perform editing on it. But due to the limitation of available paired 3D editing training data, they can not achieve satisfactory results. Recent flow-based native 3D generation models [30, 59] provide the potential of directly editing in the 3D space. And we decide to choose one of them [59] as our baseline to avoid the problems of heavily relying on the 2D domain and inadequate paired 3D editing data.

3. Method

We first provide the preliminaries in Sec. 3.1. Given the original 3D asset and the editing prompt as the only input, Vinedresser3D utilizes an MLLM (Gemini-2.5-flash [2]) as its core to obtain new guidance (Sec. 3.2) and detect the intended editing region (Sec. 3.3). After that, Vinedresser3D performs 3D editing to get the new asset through an inversion-based Trellis module (Sec. 3.4) with the guidance and the editing region. The pipeline overview is in Fig. 2.

3.1. Preliminary

Trellis [59] is a flow-based native 3D generation model. It takes either a text prompt or a single image as the condition, and generate a corresponding 3D asset. Trellis employs a structured latent representation (SLAT), which represents a 3D asset as sparse voxels with their latent features. The generation pipeline is decomposed into two stages, each utilizing a specialized rectified flow model [32]. The first stage predicts the sparse voxel structure, while the second generates the detailed latent features for each voxel. The resulting SLAT can be decoded into 3D Gaussians [25] or meshes. We adopt Trellis as our baseline given its state-of-the-art performance in high-fidelity 3D generation.

RF-Solver [53] introduces a method for the accurate inversion of rectified flow models in the context of image generation. While standard inversion typically relies on first-order discretization:

$$X_{i-1} = X_i + (t_{i-1} - t_i) v_\theta(X_i, t_i), \tag{1}$$

where X_i represents the noisy state at timestep t_i and v_θ

Change the car to a train.

Original complete description: A small, stylized toy features a glossy red and blue *car-like body* with a light tan bottom and *a blue '1' marking on its side*, *supported by four smooth light blue cylindrical wheels with orange circular centers*, carrying a white spherical figure with a yellow beak, black dot eyes, and an orange wavy crest on its head.
Names of editing part: car-like body, wheels
Edit type: Modification

New complete description: A small, stylized toy features a glossy red and blue *locomotive-style train body* with a light tan bottom and *train wheels*, carrying a white spherical figure with a yellow beak, black dot eyes, and an orange wavy crest on its head.
New complete description for stage 1: A small, stylized toy features a locomotive-style train body with a bottom and train wheels, carrying a spherical figure with a beak, dot eyes, and a wavy crest on its head.
New complete description for stage 2: A toy features a glossy red and blue train body with a light tan bottom and train wheels, carrying a white figure with a yellow beak, black eyes, and an orange crest on its head.
New parts description: Locomotive-style train body: a glossy red and blue locomotive-style train body with a light tan bottom. Train wheels: light tan train wheels.
New parts description for stage 1: ……
New parts description for stage 2: ……
……

Figure 3. Text guidance output by the MLLM. The modified words between the original complete description and the new complete description are marked with underlined italics. We highlight the extracted stage 1-related (in cyan) and stage 2-related (in red) information.

denotes the flow velocity field, RF-Solver enhances this by incorporating a second-order Taylor expansion term:

$$X_{i-1} = X_i + (t_{i-1} - t_i) v_\theta(X_i, t_i) + \frac{1}{2}(t_{i-1} - t_i)^2 v_\theta^{(1)}(X_i, t_i). \quad (2)$$

Here, $v_\theta^{(1)}$ represents the temporal derivative of the velocity field. We adopt RF-Solver in our 3D editing module because this second-order term significantly improves inversion fidelity.

3.2. Multi-modal Guidance Generation

Given the input 3D asset and a user-provided editing prompt, Vinedresser3D first generates the necessary text and image guidance to condition downstream modules. Leveraging the reasoning capabilities of a Multimodal Large Language Model (MLLM), the agent semantically understand the editing request to generate guidance signals that are both comprehensive and precisely aligned with the target modification.

Vinedresser3D first generates the text guidance (Fig. 3). The process begins with rendering multi-view images of the original 3D asset. These images, combined with the user's editing prompt, are provided as input to the MLLM. We employ a multi-step prompting strategy: first, the MLLM analyzes the input to output a description of the original asset, identify the specific names of parts targeted for editing, and classify the editing operation type (addition, modification, or deletion). Then, the model is prompted to predict the complete description of the asset post-editing. Importantly, we constrain the MLLM to maximize semantic preservation of the original description outside the targeted edit regions to retrain fine-grained details. Next, the MLLM isolates descriptions specifically for the newly modified target parts. Finally, to align with our two-stage generation pipeline, the MLLM decomposes these descriptions into structure-related components (for stage 1 geometry) and appearance-related components (for stage 2 latent features). This multi-stage process provides highly targeted conditioning for subsequent modules. As shown in Fig. 3, the MLLM demonstrates implicit 3D semantic understanding despite its 2D training, accurately processing spatial information.

Furthermore, Vinedresser3D generates image guidance through a view-selection and editing pipeline. The MLLM evaluates the rendered views to identify the candidate that maximizes the visibility of the target editing parts and the overall asset structure. The selected image is then forwarded to an image editing model (Nano Banana [1]). We augment the conditioning of the image editing model by supplying both the editing prompt and the decoupled description of the new target parts. This multi-level text conditioning guides the generation of a high-fidelity reference image that accurately reflects the desired edits.

3.3. Detect editing region

One main advantage of Vinedresser3D over previous work is that it can automatically detect the editing region, so users do not need to provide a 3D mask. Since we adopt a voxel-based 3D representation, our detection algorithm only needs to determine whether each voxel belongs to the editing region. Fig. 2 illustrates this process.

For modification and deletion requests, Vinedresser3D first identifies which parts of the original asset A should be edited and which should be preserved, denoted by P_{edit} and P_{pres}, respectively. To this end, it employs PartField [31], a powerful 3D segmentation model, to decompose A into S semantic parts. The original 3D asset and its segmented colored view (rendered as multi-view images), together with text guidance specifying the target parts, are then fed into the MLLM, which selects the editing region P_{edit} and implicitly defines the preserved region as $P_{\text{pres}} = A \setminus P_{\text{edit}}$. Since PartField's segmentation granularity depends on S, Vinedresser3D computes segmentations for multiple values of S and passes all of them to the MLLM, allowing it to choose the most fine-grained and semantically precise parts.

After partitioning the original asset A into P_{edit} and P_{pres}, Vinedresser3D must determine the corresponding editing region R_{edit} (and its complement, the preserved region

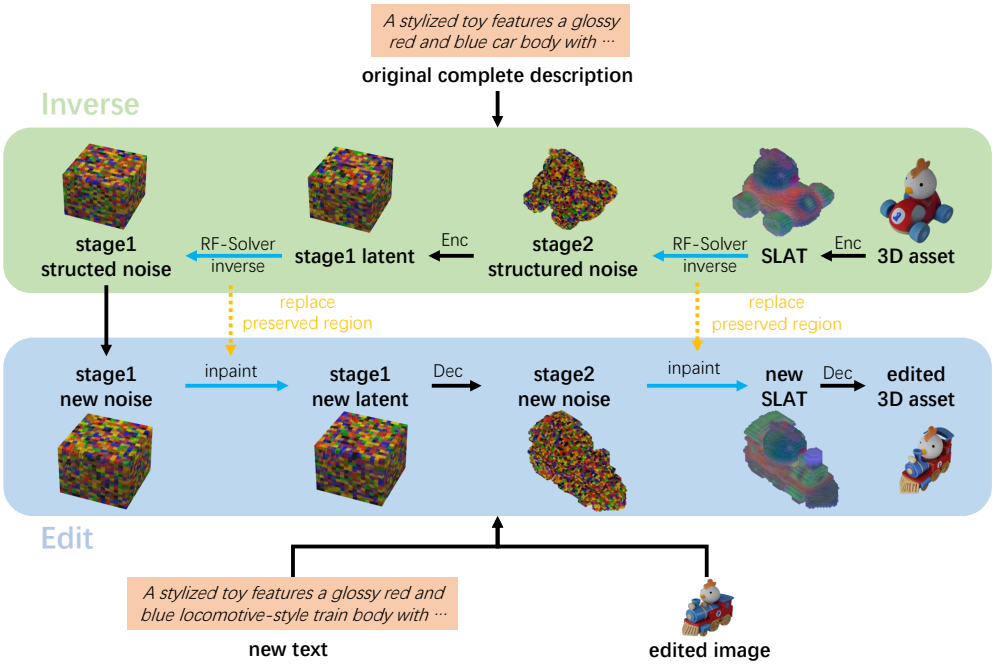

Figure 4. Our native 3D inversion-based editing pipeline. It first invert the original 3D asset back to structured noises using RF-Solver [53] and the original complete description as the condition. Then it performs editing through inpainting by denoising with Trellis-text and Trellis-image alternatively for all timesteps, using both the new text and edited image as conditions.

$R_{\text{pres}} = C \backslash R_{\text{edit}}$, where C denotes the entire voxel grid). This is straightforward for addition and deletion requests: for addition, R_{edit} is all voxels in the 3D space outside the original asset ($C \backslash A$), and for deletion, R_{edit} coincides with P_{edit}. In contrast, computing R_{edit} for modification requests is non-trivial. We cannot simply treat all voxels outside P_{pres} as editable, because Trellis may modify a layer of voxels above the preserved geometry, unintentionally altering P_{pres}.

To address this, we introduce the following procedure. We first assign all voxels in P_{edit} to R_{edit} and all voxels in P_{pres} to R_{pres}. We then compute a bounding box for each preserved semantic part and denote the union of these boxes as $bbox_{\text{pres}}$. All empty voxels outside $bbox_{\text{pres}}$ are also assigned to R_{edit}. Next, for each empty voxel v inside $bbox_{\text{pres}}$, we find its k-nearest voxels belonging to the original asset and compute the fraction of those lying in P_{edit}, denoted as $\text{PropKNN}(v)$. If this proportion exceeds a threshold τ, we classify v as editable; otherwise, it is preserved. Formally, the editing region is defined as:

$$R_{\text{edit}} = \begin{cases} C \backslash A & \text{addition} \\ P_{\text{edit}} & \text{deletion} \\ P_{\text{edit}} \cup (C \backslash bbox_{\text{pres}}) \cup V & \text{modification} \end{cases} \quad (3)$$

where $V = \{v | v \in bbox_{\text{pres}} \backslash A, \text{PropKNN}(v) > \tau\}$.

This automatic localizing pipeline equips our agent with some spatial reasoning ability. As we can see in Fig. 5, Vinedresser3D can leverage the world knowledge and common sense learnt by the MLLM to understand the user's intention and precisely locate the desired editing region in a complex 3D asset.

3.4. Inversion-Based 3D Editing

Given the multi-modal guidance and the 3D mask R_{edit}, Vinedresser3D performs localized 3D editing via an inversion-inpainting process. The asset is first inverted to its latent initial noise representation, and then edited through inpainting. We implement this through a novel inversion-based module (Fig. 4) that combines Trellis-text and Trellis-image models to achieve both high-fidelity detail and comprehensive structural coherence in the edited results.

We employ RF-Solver [53] for accurate inversion in the 3D flow models. Conditioned on the complete description of the original asset, the process begins with Stage 2 inversion to derive the structured noise for the SLATs. Then, the sparse voxel structure is encoded into the Stage 1 latent space for Stage 1 inversion, yielding the corresponding noise. Empirically, we observed through grid search that setting the Classifier-Free Guidance (CFG) [21] strength to 0 significantly stabilizes the inversion trajectory and minimizes reconstruction error. Therefore, we use a CFG scale of 0 for all inversion steps. Given the inverted noises, we implement a mask-guided inpainting strategy to synthesize the edited asset while preserving the identity of unedited regions. Specifically, at each timestep of the denoising pro-

		Text Align.	Unedited Preservation				3D Quality
Method	Human Mask	CLIP-T↑	CD↓	PSNR↑	SSIM↑	LPIPS↓	FID↓
Instant3dit [5]	✓	0.227	0.027	20.86	0.851	0.153	80.35
VoxHammer [28]	✓	0.235	0.027	24.36	0.890	0.087	34.95
Trellis [59]	✓	0.247	0.010	37.35	**0.984**	0.017	31.10
Ours	✗	**0.252**	0.016	29.45	0.953	0.045	29.49
Ours w/ HM	✓	**0.252**	**0.008**	**37.69**	**0.984**	**0.015**	**27.38**

Table 1. Quantitative comparison. We include the results of our method with human-provided 3D masks (Ours w/ HM). Best results are in **bold**. Our method with human-provided 3D masks achieves the best results in all metrics. Even though our method without human-provided 3D masks does not preserve the unedited parts as well as Trellis, it still closely aligns with the editing prompt and produces high-quality outcomes.

Model	Text Align.	Unedited Preservation	3D Quality
vs. Trellis	92.5%	82.0%	90.8%
vs. VoxHammer	89.8%	79.3%	90.2%

Table 2. User study. We ask the user to select the better one between ours and another method in terms of editing prompt alignment, unedited parts preservation and overall 3D quality. We report the win rates of our methods. Our method achieves high win rates in all perspectives.

cess, the latent features of voxels located outside the edit mask R_{edit} are replaced with their spatially corresponding features from the original inversion trajectory. Throughout this generation process, the flow models are conditioned on the decomposed Stage 1 and Stage 2 text guidance, along with the generated image guidance.

We observe that directly applying inversion-based editing with either Trellis-text or Trellis-image alone is insufficient. Trellis-text produces limited generation quality, particularly in fine details, due to the scarcity of text-aligned 3D training data [59]. Trellis-image is conditioned on a single viewpoint, and therefore struggles with occluded regions where visual information is unavailable. To address these limitations, we introduce an Interleaved Trellis editing Module that leverages the complementary strengths of both modalities. We construct the denoising trajectory by alternating between one timestep of Trellis-text denoising and one timestep of Trellis-image denoising, interleaving the vector fields of the two models. This design combines the broad semantic alignment and prompt adherence from the text branch with the high-fidelity detail from the image branch, leading to improved overall generation quality.

We now describe additional design details for mask handling across stages. Note that Trellis produces the final asset at 64^3 voxel resolution, while Stage 1 operates in a 16^3 latent space. We downscale the editing mask accordingly via proportion thresholding to obtain the Stage 1 mask. For Stage 2 editing, inspired by [28], we adopt a soft mask instead of a hard mask. Specifically, for each voxel in the preserved region, we compute its distance to the editing region. For voxels close to the boundary, we compute their features as a weighted average of the denoised and inverted features during editing, rather than directly replacing them with the inverted ones. We find that this soft masking effectively eliminates floating artifacts at the boundary of the preserved region. For deletion requests, we skip Stage 1 inversion and editing entirely. Instead, we directly remove all voxels in R_{edit} and apply Stage 2 to smooth the boundary.

4. Experiment

4.1. Setup

Implementation details. We adopt Google Gemini-2.5-Flash [2] as the MLLM and Nano Banana [1] as the image editing model. We render 8 views of each asset, for both the original appearance and the segmentation, as input to the MLLM for text guidance and editing region selection. We additionally render 24 views from varying rotations and elevations for the MLLM to select the best view as image guidance. Since PartField [31] operates on point clouds for both input and output, we decode each asset into 3D Gaussians as point clouds before feeding them into PartField, and convert the resulting segmentation back to voxels via majority voting. The number of semantic parts is set to $S \in [3, 8]$. We follow Trellis's [59] configuration for the 3D editing module. During editing, the agent explores different combinations of positive and negative prompts as text conditions and selects the best result.

Dataset. We collect high-quality and diverse 3D assets from both model-generated and human-created sources. Specifically, we gather 24 assets from Trellis generation results and 33 assets from GSO [13] and PartObjaverse-Tiny [61]. For each asset, we carefully design an editing prompt that aligns with common sense. Our prompts cover a wide range of editing categories and difficulty levels.

Baselines. We compare our method with three state-of-the-art 3D editing pipelines: Trellis [59] editing, VoxHammer [28], and Instant3dit [5]. Trellis supports 3D editing through human-provided 3D masks combined with Repaint [34]. VoxHammer builds on Trellis-image by render-

Figure 5. Qualitative comparison of different methods. We can see that our method surpasses all the others by smartly interpreting the editing intention of the user, closely following the editing prompt, precisely locating the intended editing region and generating high-fidelity results.

ing an image, applying image inpainting to obtain a new condition, and performing inversion-based 3D editing, but it still requires a human-provided 3D mask. Instant3dit is a multi-view diffusion and reconstruction-based editing model.

Metrics. We use CLIP-T [44] to measure alignment with the editing prompt, Chamfer Distance [15], PSNR, SSIM [55], and LPIPS [67] to assess preservation of unedited regions, and FID [20] to evaluate overall 3D quality. We also conduct a user study to evaluate human preference.

4.2. Qualitative Results

We present qualitative comparisons in Fig. 5, where our method surpasses all baselines. It accurately interprets the user's editing intention and faithfully follows the editing prompt to produce expected outcomes, demonstrating the effectiveness of the MLLM and the guidance it generates. Additionally, our method precisely localizes the target editing region within complex assets given only text prompts, validating the editing region detection pipeline. The overall quality of the assets is also well preserved, confirming the competence of the inversion-based Trellis editing module.

4.3. Quantitative Results

We present quantitative comparisons in Tab. 1. Since all baselines require human-provided 3D masks, we also include results of our method with human-provided masks for fair comparison. Both variants of our method, with and without human-provided masks, achieve the best CLIP-T score. This validates the benefit of leveraging an MLLM to produce detailed text guidance. For unedited region preservation metrics (*i.e.* CD, PSNR, SSIM, LPIPS), our method remains competitive even without human provided masks. Given human-provided masks, our approach achieves the best results. Finally, both settings of our method outperform all baselines in overall 3D quality (FID).

We also conduct a user study to evaluate human preference across methods. Users are asked to choose the better output between two results from three perspectives: editing

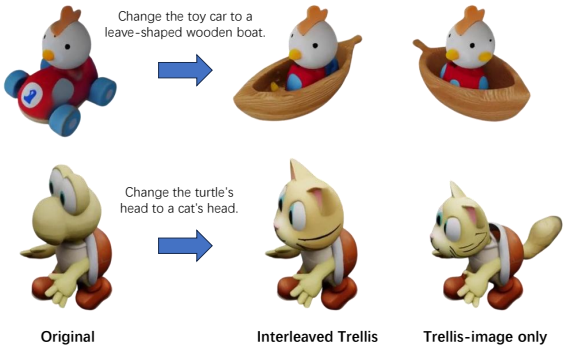

Figure 6. Qualitative ablation of interleaved Trellis vs Trellis-image only. The outputs of Trellis-image only editing may be distorted or unreasonable.

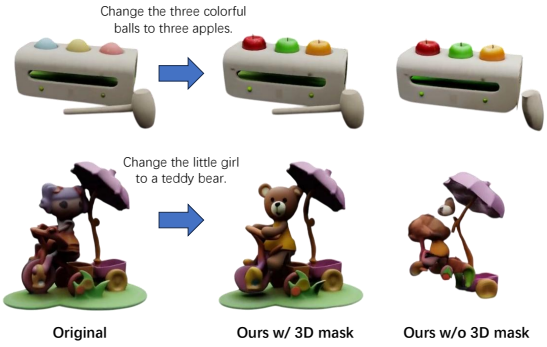

Figure 7. Qualitative ablation of our method with and without the detected editing region as the 3D mask. Our method without the detected editing region may alter the intended preserved parts or produce distorted outputs.

	Unedited Preservation			3D Quality
Methods	PSNR↑	SSIM↑	LPIPS↓	FID↓
Ours	**29.45**	**0.953**	**0.045**	**29.49**
Ours w/o Trellis-text	28.06	0.943	0.054	30.59
Ours w/o R_{edit}	25.65	0.921	0.068	33.95

Table 3. Quantitative results of the ablation study. We compare our method with the Trellis-image only 3D editing design and ours without the detected editing region. We use PSNR, SSIM and LPIPS to measure unedited parts preservation and FID to measure the overall 3D quality. We can see that our method outperforms both of the ablated ones on all metrics.

prompt alignment, unedited region preservation, and overall 3D quality. The win rates are reported in Tab. 2. The results clearly show that our method consistently outperforms both Trellis and VoxHammer across all perspectives.

4.4. Ablation Study

We conduct two ablation experiments to validate key components of our agent. The first ablation uses only Trellis-image in the editing module, verifying the effect of the interleaved Trellis design. The second ablation removes the editing region mask (*i.e.*, performing purely interleaved Trellis generation instead of inpainting), verifying the necessity of the editing region detection module.

When using only Trellis-image, the editing module receives limited information due to occlusion and deformation during rendering, which can lead to distorted or unreasonable outputs. We provide qualitative examples in Fig. 6 and report FID scores in Tab. 3, demonstrating the importance of interleaving Trellis-text and Trellis-image for overall 3D quality.

In the second ablation, we verify that providing the editing region to the editing module helps preserve unedited parts (the first example in Fig. 7 and PSNR, SSIM, LPIPS in Tab. 3). It also plays an important role in maintaining the overall quality of the edited asset (the second example in Fig. 7 and FID in Tab. 3). Without a mask, the interleaved Trellis editing can fail to handle the asset properly and produce distorted outputs. Providing a 3D mask and injecting preserved-region features during editing regularizes the denoising process and helps maintain overall quality.

5. Limitation

While Vinedresser3D can produce impressive results, it still has several limitations. First, the underlying MLLM does not accept native 3D input. We expect that enabling the MLLM to directly consume 3D inputs and perform 3D reasoning could further improve the results. Second, the external tools invoked by Vinedresser3D are imperfect. For example, PartField [31] can produce unreasonable part segmentations. We anticipate that future advances in these 3D models will further improve the overall performance of our agent.

6. Conclusion

We develop an agent that is capable of intelligently and efficiently performing high-quality text-guided 3D editing. Using an MLLM as its core, it smartly makes use of an image editing model, a 3D segmentation model and a 3D generation baseline. It intelligently interprets the editing request, generates detailed and comprehensive guidance, automatically detects the intended editing region and performs precise editing. It produce high-quality results in terms of editing prompt alignment, unedited parts preservation and overall 3D quality. Also, we show the potential of integrating 2D MLLMs into 3D pipelines. We believe our work will promote the 3D editing field towards an agentic, high-quality and smart future.

References

[1] Introducing gemini 2.5 flash image, our state-of-the-art image model. https://developers.googleblog.com/en/introducing-gemini-2-5-flash-image/, 2025. Google Developers Blog.

[2] Gemini-2.5-flash. https://deepmind.google/models/gemini/flash/, 2025. Google DeepMind.

[3] Panos Achlioptas, Ian Huang, Minhyuk Sung, Sergey Tulyakov, and Leonidas Guibas. Shapetalk: A language dataset and framework for 3d shape edits and deformations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12685–12694, 2023.

[4] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the International Conference on Learning Representations*, 2024.

[5] Amir Barda, Matheus Gadelha, Vladimir G Kim, Noam Aigerman, Amit H Bermano, and Thibault Groueix. Instant3dit: Multiview inpainting for fast editing of 3d objects. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16273–16282, 2025.

[6] Siyuan Bian, Chenghao Xu, Yuliang Xiu, Artur Grigorev, Zhen Liu, Cewu Lu, Michael J Black, and Yao Feng. Chatgarment: Garment estimation, generation and editing via large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2924–2934, 2025.

[7] Hansheng Chen, Ruoxi Shi, Yulin Liu, Bokui Shen, Jiayuan Gu, Gordon Wetzstein, Hao Su, and Leonidas Guibas. Generic 3d diffusion adapter using controlled multi-view editing. *arXiv preprint arXiv:2403.12032*, 2024.

[8] Junlong Chen, Jens Grubert, and Per Ola Kristensson. Analyzing multimodal interaction strategies for llm-assisted manipulation of 3d scenes. In *2025 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 206–216. IEEE, 2025.

[9] Minghao Chen, Junyu Xie, Iro Laina, and Andrea Vedaldi. Shap-editor: Instruction-guided latent 3d editing in seconds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26456–26466, 2024.

[10] Yige Chen, Teng Hu, Yizhe Tang, Siyuan Chen, Ang Chen, and Ran Yi. Plasticine3d: 3d non-rigid editing with text guidance by multi-view embedding optimization. *arXiv preprint arXiv:2312.10111*, 2023.

[11] Dale Decatur, Itai Lang, Kfir Aberman, and Rana Hanocka. 3d paintbrush: Local stylization of 3d shapes with cascaded score distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4473–4483, 2024.

[12] Shaocong Dong, Lihe Ding, Zhanpeng Huang, Zibin Wang, Tianfan Xue, and Dan Xu. Interactive3d: Create what you want by interactive 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4999–5008, 2024.

[13] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022.

[14] Ziya Erkoç, Can Gümeli, Chaoyang Wang, Matthias Nießner, Angela Dai, Peter Wonka, Hsin-Ying Lee, and Peiye Zhuang. Preditor3d: Fast and precise 3d shape editing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 640–649, 2025.

[15] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[16] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. In *Advances in Neural Information Processing Systems*, pages 18225–18250, 2023.

[17] Will Gao, Dilin Wang, Yuchen Fan, Aljaz Bozic, Tuur Stuyck, Zhengqin Li, Zhao Dong, Rakesh Ranjan, and Nikolaos Sarafianos. 3d mesh editing using masked lrms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7165, 2025.

[18] Zeqi Gu, Yin Cui, Zhaoshuo Li, Fangyin Wei, Yunhao Ge, Jinwei Gu, Ming-Yu Liu, Abe Davis, and Yifan Ding. Artiscene: Language-driven artistic 3d scene generation through image intermediary. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2891–2901, 2025.

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

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, 2017.

[21] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[22] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. In *Advances in Neural Information Processing Systems*, pages 20482–20494, 2023.

[23] Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *Proceedings of the International Conference on Machine Learning*, 2024.

[24] Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. In *European Conference on Computer Vision*, pages 297–314. Springer, 2024.

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time

radiance field rendering. *ACM Transactions on Graphics*, 42 (4):139–1, 2023.

[26] Umar Khalid, Hasan Iqbal, Azib Farooq, Jing Hua, and Chen Chen. 3dego: 3d editing on the go! In *European Conference on Computer Vision*, pages 73–89. Springer, 2024.

[27] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2426–2435, 2022.

[28] Lin Li, Zehuan Huang, Haoran Feng, Gengxiong Zhuang, Rui Chen, Chunchao Guo, and Lu Sheng. Voxhammer: Training-free precise and coherent 3d editing in native 3d space. *arXiv preprint arXiv:2508.19247*, 2025.

[29] Yuhan Li, Yishun Dou, Yue Shi, Yu Lei, Xuanhong Chen, Yi Zhang, Peng Zhou, and Bingbing Ni. Focaldreamer: Text-driven 3d editing via focal-fusion assembly. In *Proceedings of the AAAI conference on artificial intelligence*, pages 3279–3287, 2024.

[30] Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, et al. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. *arXiv preprint arXiv:2502.06608*, 2025.

[31] Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9704–9715, 2025.

[32] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *Proceedings of the International Conference on Learning Representations*, 2023.

[33] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[34] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[35] Xianzheng Ma, Yash Bhalgat, Brandon Smart, Shuai Chen, Xinghui Li, Jian Ding, Jindong Gu, Dave Zhenyu Chen, Songyou Peng, Jia-Wang Bian, et al. When llms step into the 3d world: A survey and meta-analysis of 3d tasks via multi-modal large language models. *arXiv preprint arXiv:2405.10255*, 2024.

[36] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, pages 46534–46594, 2023.

[37] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *Proceedings of the International Conference on Learning Representations*, 2022.

[38] Daiki Miyake, Akihiro Iohara, Yu Saito, and Toshiyuki Tanaka. Negative-prompt inversion: Fast image inversion for editing with text-guided diffusion models. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2063–2072. IEEE, 2025.

[39] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6038–6047, 2023.

[40] Başak Melis Öcal, Maxim Tatarchenko, Sezer Karaoğlu, and Theo Gevers. Sceneteller: Language-to-3d scene generation. In *European Conference on Computer Vision*, pages 362–378. Springer, 2024.

[41] Francesco Palandra, Andrea Sanchietti, Daniele Baieri, and Emanuele Rodola. Gsedit: Efficient text-guided editing of 3d objects via gaussian splatting. *arXiv preprint arXiv:2403.05154*, 2024.

[42] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *Proceedings of the International Conference on Learning Representations*, 2023.

[43] Zhangyang Qi, Yunhan Yang, Mengchen Zhang, Long Xing, Xiaoyang Wu, Tong Wu, Dahua Lin, Xihui Liu, Jiaqi Wang, and Hengshuang Zhao. Tailor3d: Customized 3d assets editing and generation with dual-side images. *arXiv preprint arXiv:2407.06191*, 2024.

[44] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[45] Litu Rout, Yujia Chen, Nataniel Ruiz, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Semantic image inversion and editing using rectified stochastic differential equations. In *Proceedings of the International Conference on Learning Representations*, 2025.

[46] Aadarsh Sahoo, Vansh Tibrewal, and Georgia Gkioxari. Aligning text, images, and 3d structure token-by-token. *arXiv preprint arXiv:2506.08002*, 2025.

[47] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 430–440, 2023.

[48] Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. Assisting in writing wikipedia-like articles from scratch with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6252–6278, 2024.

[49] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning,

2023. In *Advances in neural information processing systems*, 2023.

[50] Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3d-gpt: Procedural 3d modeling with large language models. In *2025 International Conference on 3D Vision (3DV)*, pages 1253–1263. IEEE, 2025.

[51] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1921–1930, 2023.

[52] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20902–20911, 2024.

[53] Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. Taming rectified flow for inversion and editing. In *Proceedings of the International Conference on Machine Learning*, 2025.

[54] Yuxuan Wang, Xuanyu Yi, Zike Wu, Na Zhao, Long Chen, and Hanwang Zhang. View-consistent 3d editing with gaussian splatting. In *European conference on computer vision*, pages 404–420. Springer, 2024.

[55] Z Wang. Image quality assessment: Form error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):604–606, 2004.

[56] Zhenwei Wang, Tengfei Wang, Zexin He, Gerhard Hancke, Ziwei Liu, and Rynson WH Lau. Phidias: A generative model for creating 3d content from text, image, and 3d conditions with reference-augmented diffusion. In *Proceedings of the International Conference on Learning Representations*, 2025.

[57] Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis Ioannidis, Karthik Subbian, Jure Leskovec, and James Y Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. In *Advances in Neural Information Processing Systems*, pages 25981–26010, 2024.

[58] Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. Collabllm: From passive responders to active collaborators. In *Proceedings of the International Conference on Machine Learning*, 2025.

[59] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025.

[60] Jiale Xu, Xintao Wang, Yan-Pei Cao, Weihao Cheng, Ying Shan, and Shenghua Gao. Instructp2p: Learning to edit 3d point clouds with text instructions. *arXiv preprint arXiv:2306.07154*, 2023.

[61] Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y Lam, Yan-Pei Cao, and Xihui Liu. Sampart3d: Segment any part in 3d objects. *arXiv preprint arXiv:2411.07184*, 2024.

[62] Kaixin Yao, Longwen Zhang, Xinhao Yan, Yan Zeng, Qixuan Zhang, Lan Xu, Wei Yang, Jiayuan Gu, and Jingyi Yu. Cast: Component-aligned 3d scene reconstruction from an rgb image. *ACM Transactions on Graphics (TOG)*, 44(4): 1–19, 2025.

[63] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations*, 2022.

[64] Junliang Ye, Zhengyi Wang, Ruowen Zhao, Shenghao Xie, and Jun Zhu. Shapellm-omni: A native multimodal llm for 3d generation and understanding. In *Advances in neural information processing systems*, 2025.

[65] Fukun Yin, Xin Chen, Chi Zhang, Biao Jiang, Zibo Zhao, Wen Liu, Gang Yu, and Tao Chen. Shapegpt: 3d shape generation with a unified multi-modal language model. *IEEE Transactions on Multimedia*, 2025.

[66] Qihang Zhang, Chaoyang Wang, Aliaksandr Siarohin, Peiye Zhuang, Yinghao Xu, Ceyuan Yang, Dahua Lin, Bolei Zhou, Sergey Tulyakov, and Hsin-Ying Lee. Towards text-guided 3d scene composition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6829–6838, 2024.

[67] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[68] Jingyu Zhuang, Di Kang, Yan-Pei Cao, Guanbin Li, Liang Lin, and Ying Shan. Tip-editor: An accurate 3d editor following both text-prompts and image-prompts. *ACM Transactions on Graphics (TOG)*, 43(4):1–12, 2024.