SplatPainter: Interactive Authoring of 3D Gaussians from 2D Edits via Test-Time Training

Yang Zheng[1,2] Hao Tan[2] Kai Zhang[2] Peng Wang[2]

Leonidas Guibas[1] Gordon Wetzstein[1] Wang Yifan[2]

[1]Stanford University [2]Adobe Research

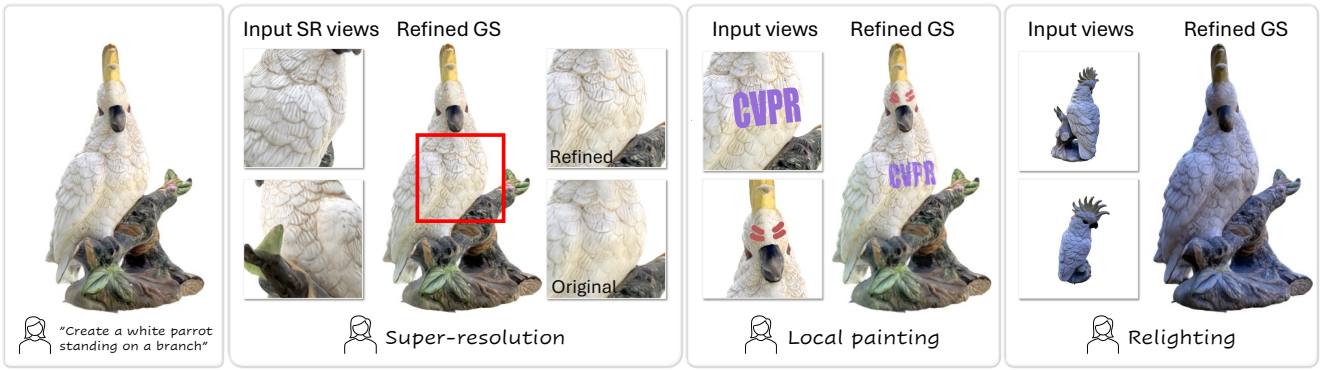

Figure 1. We present SplatPainter, a feedforward method to support interactive and continuous authorship of 3D Gaussian assets through intuitive 2D edits *e.g.* high-fidelity local detail refinement, local paint-over, and consistent global recoloring, while maintaining 3D consistency and the original texture and structure details.

Abstract

The rise of 3D Gaussian Splatting has revolutionized photorealistic 3D asset creation, yet a critical gap remains for their interactive refinement and editing. Existing approaches based on diffusion or optimization are ill-suited for this task, as they are often prohibitively slow, destructive to the original asset's identity, or lack the precision for fine-grained control. To address this, we introduce SplatPainter, a state-aware feedforward model that enables continuous editing of 3D Gaussian assets from user-provided 2D view(s). Our method directly predicts updates to the attributes of a compact, feature-rich Gaussian representation and leverages Test-Time Training to create a state-aware, iterative workflow. The versatility of our approach allows a single architecture to perform diverse tasks, including high-fidelity local detail refinement, local paint-over, and consistent global recoloring, all at interactive speeds, paving the way for fluid and intuitive 3D content authoring. Our project website is at: https://y-zheng18.github.io/SplatPainter/.

1. Introduction

High-quality 3D content creation is fundamental to a growing number of industries, from gaming and virtual reality to e-commerce and digital twins, and is central to the future of digital interaction. Recently, 3D Gaussian Splatting (3DGS) [39] has emerged as a leading representation for its ability to render photorealistic imagery in real-time, while offering a flexible topology that captures complex details like hair and fur more effectively than traditional meshes. As text- and image-to-3D pipelines [43, 75] built on GS become more powerful, they enable the rapid creation of assets. However, these one-shot generation methods often represent a starting point rather than a final product. Their focus on global coherence frequently comes at the cost of local detail, creating a crucial need for an efficient and intuitive workflow to refine these assets post-creation.

The critical missing piece in the modern 3D workflow is a framework for interactively and intuitively editing existing 3DGS assets. For content creation to be truly powerful, users need to make precise adjustments with familiar 2D tools, applying edits to arbitrary views, including close-ups for fine-grained refinement. This is a non-trivial challenge. Any solution must address three core difficulties: (1) propagating a 2D edit consistently across the 3D asset, (2) preserving the identity and intricate details of unedited regions, and (3) operating at interactive speeds for a fluid, iterative creative process. Solving these problems is vital, as it would unlock a true end-to-end content creation pipeline, giving

artists direct, fine-grained control and saving time and compute compared to wrestling with prompts for re-generation.

Current 3D editing approaches are ill-suited for this task of interactive refinement. Diffusion-based methods [2, 3, 6, 11, 15, 23, 42, 54, 81] leverage powerful generative priors but are fundamentally destructive; they typically generate new multiview images and reconstruct the asset from scratch, corrupting the original content. Furthermore, they are designed for large, stylistic changes via text prompts and lack the precision for pixel-level edits, while their iterative sampling process is far too slow for real-time feedback. On the other hand, optimization-based refinement techniques [25, 41, 62, 72, 82, 90] can preserve asset identity but are prohibitively slow—often taking hundreds of seconds for a single change—rendering them impractical for interactive use. These methods also struggle when provided with only a few sparse input views, a common artistic workflow. While one might consider converting the asset to a mesh for editing, this process is notoriously lossy, often failing to preserve the view-dependent effects and fine structures that make Gaussian Splatting so compelling.

In this work, we introduce SplatPainter, a novel framework for fast, precise, and identity-preserving editing of 3D Gaussian assets. Our core contribution is a state-aware feedforward model that takes an existing GS asset and a 2D image-space edit, and directly predicts the updates to individual Gaussian attributes. This feedforward design bypasses slow optimization or generation loops, enabling interactive performance. To support a continuous, iterative workflow, our model is made state-aware via Test-Time Training (TTT) [65, 78, 97], allowing it to adapt to user edits on the fly. This process is built upon a compact, feature-rich GS representation created in an efficient, one-time preprocessing step that both regularizes geometry and enriches each Gaussian with the global context needed for intelligent editing. The versatility of our architecture allows it to handle a wide range of tasks within a single, unified framework, from local detail refinement to consistent global recoloring.

Our main contributions can be summarized as follows:

1. **A Novel Feedforward Editing Framework:** We introduce SplatPainter, a state-aware feedforward model for the interactive and continuous editing of 3DGS. By leveraging TTT, our method iteratively incorporates user edits from 2D views, directly updating the attributes of a compact and feature-rich Gaussian representation without destructive regeneration.
2. **Versatile and Intuitive Applications:** We evaluate our framework on two novel and practical applications: high-fidelity local detail refinement and consistent global relighting. The unified approach provides users with precise control over both fine-grained texture details and broad appearance changes from 2D edit(s).
3. **State-of-the-Art Performance at Interactive Speeds:** Through extensive experiments, we show that SplatPainter significantly outperforms existing methods in both quality and speed. Our approach achieves superior visual results while operating at interactive rates, making it a truly practical solution for unlocking fluid, real-time creative workflows in 3D content authoring.

2. Related Work

Our work lies at the intersection of editing for modern 3D representations like NeRFs [48] and 3D Gaussian Splatting (3DGS) [39], and continuous 3D reconstruction. We begin by reviewing the evolving ecosystem of direct GS editing before discussing broader paradigms for image-driven 3D editing and the architectural principles that enable our iterative approach.

The Evolving 3D Gaussian Splatting Ecosystem. The wide adoption of 3DGS [39] and its plethora of extensions [24] has propelled research into making GS assets directly editable, leading to the emergence of interactive tools for manual selection and editing [34, 45, 46, 52]. More advanced methods achieve real-time geometric deformation through auxiliary cage structures [33, 68, 86] or use feedforward networks for tasks like splat densification [13, 50]. Other works augment the GS representation itself to enable new modalities like physics simulation [87]. While these approaches showcase a clear demand for direct and efficient GS manipulation, they primarily focus on geometry modification. They do not provide a mechanism for our task: intuitive, fine-grained appearance editing guided by 2D image inputs.

Image-Driven 3D Editing. Image-driven 3D editing is a central challenge, with two dominant but slow strategies emerging. The first, an optimization-heavy approach, updates a 3D representation by optimizing it to match a set of edited 2D views. This paradigm is used for tasks like geometry and texture refinement [14, 56], super-resolution [25, 41, 62, 72, 82, 83, 85, 90, 91], and CLIP-guided stylization [10, 28, 31, 40, 47, 55, 63, 73, 74, 76, 100]. Score Distillation Sampling [53] is a popular variant, where optimization is guided by text prompts instead of explicit images[8, 18, 20, 21, 36, 44, 51, 60, 71, 88, 93, 102, 103]. Although these methods can produce high-quality results, their reliance on slow, per-scene optimization makes them unsuitable for interactive workflows. The second strategy, generative reconstruction, uses a multi-view diffusion model to generate consistent 2D edits from a prompt [2, 3, 6, 7, 11, 15, 23, 30, 42, 54, 69, 80, 81], which are then lifted to a new 3D asset using a feedforward reconstruction model [32, 89, 94]. This workflow, while faster than full optimization, is still too slow for real-time use. It is also inherently destructive, as it replaces the original asset, and is limited by the diffusion model's resolution. Our method, in contrast to both strategies, is fast, non-destructive, and

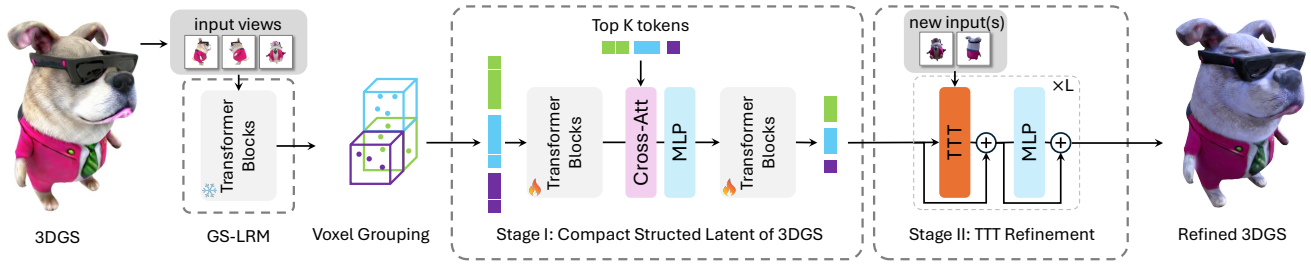

Figure 2. **Overview.** Given an input 3DGS asset, the framework first performs a one-time preprocessing step. The asset is rendered from multiple views to generate feature-rich inputs from a Gaussian LRM. Stage I compresses this into a compact latent representation via a local transformer. The interactive editing loop in Stage II then iteratively refines this latent representation using new 2D user edits (New input(s)) and a Test-Time Training (TTT) module to produce the final edited 3DGS.

handles fine-grained image edits.

Continuous and State-Aware Architectures. Our architecture is inspired by continuous reconstruction, where a 3D representation is progressively improved with new observations. This concept spans classical SLAM [22, 26, 49, 67, 101] and Structure-from-Motion [1, 58] to modern learning-based methods using recurrent or memory-augmented networks to maintain state [16, 29, 35, 37, 59, 64, 98]. Recent frameworks explicitly maintain latent states for continuous updates [9, 77, 79], while in the 3DGS domain, this is often done via continual optimization [92]. Drawing from these state-aware designs, we use test-time training (TTT) [97] to enable our feedforward model to iteratively incorporate user edits, achieving an efficient, online refinement workflow.

3. Method

Given a 3DGS asset $\mathcal{G} = \{g_i\}_{i=1}^{N}$, our goal is to efficiently update Gaussian attributes—position, scale, opacity, rotation, and spherical harmonics—based on user-provided image-space edits such as zoom-in super-resolution, graffiti-style markings, or global recoloring (see Fig. 1). Our key contribution is a feedforward, state-aware pipeline composed of two key components: (1) a compact and feature-rich Gaussian representation extracted via a one-time preprocessing step, and (2) a refinement module with Test-Time Training (TTT) layers [97] that incorporate new user inputs to iteratively update the scene (Fig. 2).

3.1. Compact and Feature-Rich 3DGS

Intelligently propagating a 2D edit requires a 3D representation with rich global context. Our strategy is to leverage a Gaussian large reconstruction model (LRM) [12, 89, 94] to create a high-fidelity, identity-preserving copy of the asset that is inherently enriched with such features. To accomplish this, we first render the input 3DGS from a set of canonical viewpoints. These rendered images serve as the input for the LRM, which then processes them to generate a new, dense cloud of Gaussians where each point is associated with a rich feature vector reflecting the global context. While this provides the necessary context, the LRM's native output is a dense, unstructured cloud of Gaussians—unsuitable for interactive editing due to high redundancy and memory overhead. Therefore, we introduce a one-time preprocessing step: a voxel-based transformer that compresses and regularizes this dense output. This process distills the globally-informed features into a compact, structured representation ideal for efficient, iterative refinement.

Preliminary: Gaussian LRM. Recent Gaussian LRMs provide a strong foundation for generating high-quality Gaussian Splatting assets from sparse multi-view images. A common approach in these frameworks is to process input multiview images $\{I_v\}_{v=1}^{V}$ with a transformer-based backbone. This backbone typically extract image tokens from patchified multiview images and then feed the tokens through a chain of self-attention transformer blocks to obtain feature tokens $\{F_v\}_{v=1}^{n}$. These tokens are then decoded to predict per-pixel Gaussian attributes g_i. Our work builds on this architectural paradigm to acquire the initial feature-rich Gaussians before our compression stage.

Voxelization. To impose a 3D structure, we first reorganize the reconstructed Gaussians into a regular latent grid. We voxelize the scene, grouping nearby Gaussians into local cells. Each resulting voxel $\mathcal{V} = \{(f_k, g_k)\}_{k=1}^{N_k}$ contains N_k Gaussians, where each Gaussian is associated with its feature token f_k from the LRM backbone and its Gaussians attributes g_k.

Local Voxel Transformer. On top of this structured representation, we introduce a local voxel transformer to produce compact and context-aware latent representations for Gaussians in each voxel. This is a two-stage process of aggregation and distillation. Given a voxel $\mathcal{V} = \{(f_k, g_k)\}_{k=1}^{N_k}$, we first aggregate intra-voxel relations via a transformer encoder consisting of L_{enc} layers of self-attention [70]:

$$\mathcal{Z} = \text{TransformerEncoder}(\mathcal{V}), \tag{1}$$

where $\mathcal{Z} = \{z_k\}_{k=1}^{N_k}$ is the set of context-aware latent codes for each Gaussian within the voxel. To reduce redundancy and summarize this information, we select the top-K most

significant Gaussians based on their opacity α and use their latent codes as queries in a cross-attention mechanism:

$$\tilde{\mathcal{Z}} = \text{CrossAttn}\big(\{z_i\}_{i\in \text{TopK}_\alpha}, \mathcal{Z}\big). \tag{2}$$

This step effectively distills the information from all Gaussians in the voxel into a compact set of features. Finally, a transformer decoder with L_{dec} self-attention layers refines these compressed features into the final, compact voxel latent:

$$\tilde{\mathcal{V}} = \text{TransformerDecoder}(\tilde{\mathcal{Z}}). \tag{3}$$

The resulting of voxel latents $\{\tilde{\mathcal{V}}\}$ provide a structured and memory-efficient representation ready for downstream refinement. In practice, we find $K = \max(\lfloor 0.25N_k \rfloor, 1)$, *i.e.* keeping up to 25% Gaussians per voxel provides sufficient compression without inducing quality degradation.

3.2. Refinement via Test-Time Training (TTT)

To enable iterative editing, our refinement module must adapt to new user inputs on the fly without overwriting the knowledge of the original asset. We achieve this using Test-Time Training (TTT) [4, 5, 38, 65, 66, 78, 97], which updates a small set of "fast weights" [57] while keeping the main network backbone frozen. This allows our model to become state-aware, integrating new information from a user's edit before applying it to the 3D scene.

Preliminary: Test-Time Training. Test-Time Training adapts a model to a new test sample by defining a self-supervised task at inference time. Within a transformer layer, input tokens are projected into queries (q), keys (k), and values (v) by linear layers with "slow weights". TTT introduces a small, adaptable mapping network, f_W, governed by a set of "fast weights" W. For a given test sample, these fast weights are updated via gradient descent on a self-supervised loss:

$$\mathcal{L}_{\text{adapt}} = \mathbb{E}_{(k,v)\sim\text{TestSample}}\left[\ell_2(f_W(k), v)\right], \tag{4}$$

where the goal is for f_W to learn to reconstruct the value from the key for pairs drawn from the test sample. Once adapted, the module processes queries from the same sample to produce the final output $o = f_W(q)$. This mechanism separates stable, general knowledge (slow weights) from instance-specific adaptation (fast weights).

Gaussian Refinement via TTT. Our refinement module is a deep transformer composed of a series of blocks, where each block contains a TTT layer followed by a feedforward network (MLP), as illustrated in Fig. 2. Within each TTT layer, the model first adapts to the user's edit and then applies this adaptation in parallel to both the image and the voxel latents.

First, the fast weights W of the mapping network f_W are updated. The set of image tokens from all new user views,

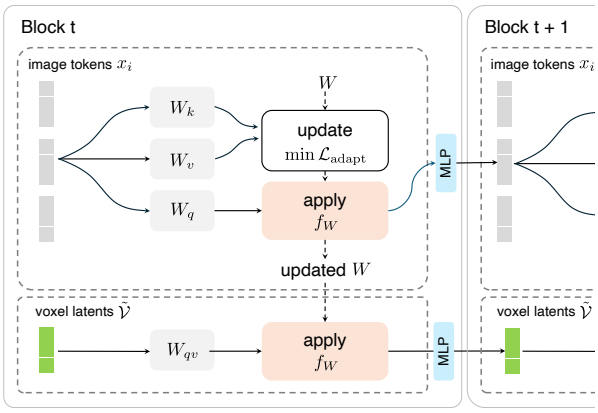

Figure 3. **TTT operations.** Fast weights W are iteratively updated using new input views and subsequently applied to the voxel GS latents after seeing all the new inputs. The residual connections are omitted for clarity.

$X_{img} = \{x_i\}$, are projected into key and value vectors using frozen (slow) projection heads, W_k and W_v:

$$k_i = W_k x_i, \quad v_i = W_v x_i, \quad \text{for } x_i \in X_{img}. \tag{5}$$

The fast weights W are updated by minimizing Eq. (4)) over all these (k_i, v_i) pairs. This single update step infuses the TTT layer with the information from the entire user edit.

Once the fast weights are updated, the adapted mapping network f_W is applied to two parallel processing streams. For the image stream, the image tokens are projected into queries using a slow head W_q, and the adapted network produces updated image tokens:

$$x_i \leftarrow f_W(W_q x_i). \tag{6}$$

For the 3D scene stream, the voxel latents $\tilde{\mathcal{V}}$ are projected into queries using a separate slow head W_{qv}, and the same adapted network produces the updated voxel latents:

$$\tilde{\mathcal{V}} \leftarrow f_W(W_{qv}\tilde{\mathcal{V}}). \tag{7}$$

The outputs of both streams are then passed through their respective MLPs. The entire block is repeated multiple times, allowing the model to progressively refine the 3D asset by repeatedly conditioning on the user's input. Following recent work [97], we implement the parametric mapping f_W as a SwiGLU-MLP [61].

3.3. Applications

The unified design of our framework supports a wide range of editing tasks:

- **Local Edits.** Our method enables refinement of high-frequency details from sparse close-up views (super-resolution) and integration of user-specified patches such as graffiti or text edits (see Fig. 1). These edits are applied to the visible voxels corresponding to the edited input views. When a new view is provided, we perform

ray tracing to identify voxels that first intersect with the camera rays, ensuring updates are localized to intended region and the rest of the details are naturally preserved. In the local region, we further update the original GS tokens using refined tokens by adding an extra TTT layer, then merge the two token sets—effectively doubling the number of Gaussians to introduce additional fine-grained structures. Please refer to the supplementary material for additional implementation details.

- **Global Edits.** In addition to local refinement, our network also supports consistent global modifications. By applying the updated fast weights from TTT layers to all Gaussian tokens, our method enables coherent changes to appearance attributes across the entire scene. This design allows users to perform holistic edits while preserving geometry and fine structures of the original asset. We demonstrate relighting as a representative example.

Together, these components constitute SplatPainter: a feed-forward, state-aware framework for interactive 3DGS refinement. Both local and global editing modules are trained with an image-based reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \|I_{\text{render}} - I_{\text{gt}}\|_2 + \lambda_{\text{perc}}\|\phi(I_{\text{render}}) - \phi(I_{\text{gt}})\|_2, \quad (8)$$

where $I_{\text{render}} = R(\text{MLP}(\tilde{\mathcal{V}}), P_v)$ denotes the rendered image from the refined Gaussians decoded from $\tilde{\mathcal{V}}$ under camera pose P_v, and $\phi(\cdot)$ is the LPIPS feature extractor [95].

4. Experiment

In this section, we explain our experimental setup (datasets, implementation details), baselines, evaluation results, and ablation study in detail.

4.1. Setup

Data. We use Objaverse1.0 [19] and TexVerse [99] (4K and 8K subsets) for training. For the local refinement application, we render 32 2K resolution images under uniform lighting, and additionally overlay random graffiti strokes to the object texture maps. For the relighting application, we render 32 512×512 resolution images under 24 lighting conditions using randomly sampled HDRI environmental maps. The cameras are randomly placed around the object with random distances following LRM [32]. For evaluation, we sampled 370 examples from the remaining TexVerse dataset that are not used for training through auto-captioning, focusing on assets with high-resolution textures and rich high-frequency details to better showcase our model's capability in fine-grained refinement.

Implementation Details. In Stage I, the Gaussian LRM is based on GS-LRM [94], and we extended the original implementation to produce view-dependent Gaussians with 4 orders of SH. The local transformer contains six encoder self-attention transformer blocks, one cross-attention block, and six decoder self-attention transformer blocks (see Fig. 2), each of which using 512 dimensions using 8 heads implemented with Flash-Attention [17]. In Stage II, the refinement network has eight TTT layers. The fast weights are obtained by taking 5 gradient decent steps using Muon-update rule following Zhang et al. [97]. We first train the local transformer in stage 1, then freeze the local transformer and train the refinement network. Both stages are trained using image reconstruction loss as introduced in Eq. (8) with $\lambda_{perc} = 0.5$. For stage 1, we use 9 to 16 input images, and for stage 2 we train with 1 to 4 new inputs. All input images are at 512×512 resolution, downsampled with bicubic interpolation from the original rendering resolution, except for the new views in the local refinement application, where new input views are cropped from the original 2K-resolution images to simulate close-up editing. Training is conducted on 16 A100 80GB GPUs and all testing and performance report is done on one A100 80GB GPU. Please refer to the supplementary session for more details.

4.2. Evaluation On Local Editing

We evaluate the local editing capability through *local refinement* application, in which a user supplies high-resolution 2D zoom-in views to refine the 3D Gaussians.

Baselines. We find three types of existing work that can be repurposed to serve as a baseline for this task, and pick a publicly available state-of-the-art model under each category as baseline for comparison. 1. Direct reconstruction: methods that reconstruct the entire Gaussian asset given multi-view input images. We choose the retrained view-dependent GS-LRM as the representative method in this category. 2. Feedforward Gaussian upsampling: methods that update Gaussian purely based on 3D prior without input image guidance, represented by GenDen [50]. 3. Refinement through optimization: optimize the Gaussians fusing the original rendering and the new images. Here, we include the original Gaussian implementation [39] and SRGS [25]. We use 9 input views uniformly sampled around the sphere to get the compact and feature-rich 3DGS from stage I, and 2 additional zoomed-in views for refinement. All 11 views are jointly fed to GS-LRM and lifted to produce the 3DGS in one shot. For GenDen and optimization-based approaches, the input 3DGS is initialized from output of the GS-LRM model using the 9 base views, and specifically for the optimization-based approaches, 48 additional uniformly sampled views of this initial 3DGS are provided during optimization to alleviate quality degradation under the sparse view optimization.

Results. To evaluate the refinement performance, we evaluate the image quality at two novel views sampled around each zoomed-in views. As shown in Tab. 1 and Fig. 4, our method achieves the best visual and quantitative performance among all baselines. Compared to GS-LRM [94]

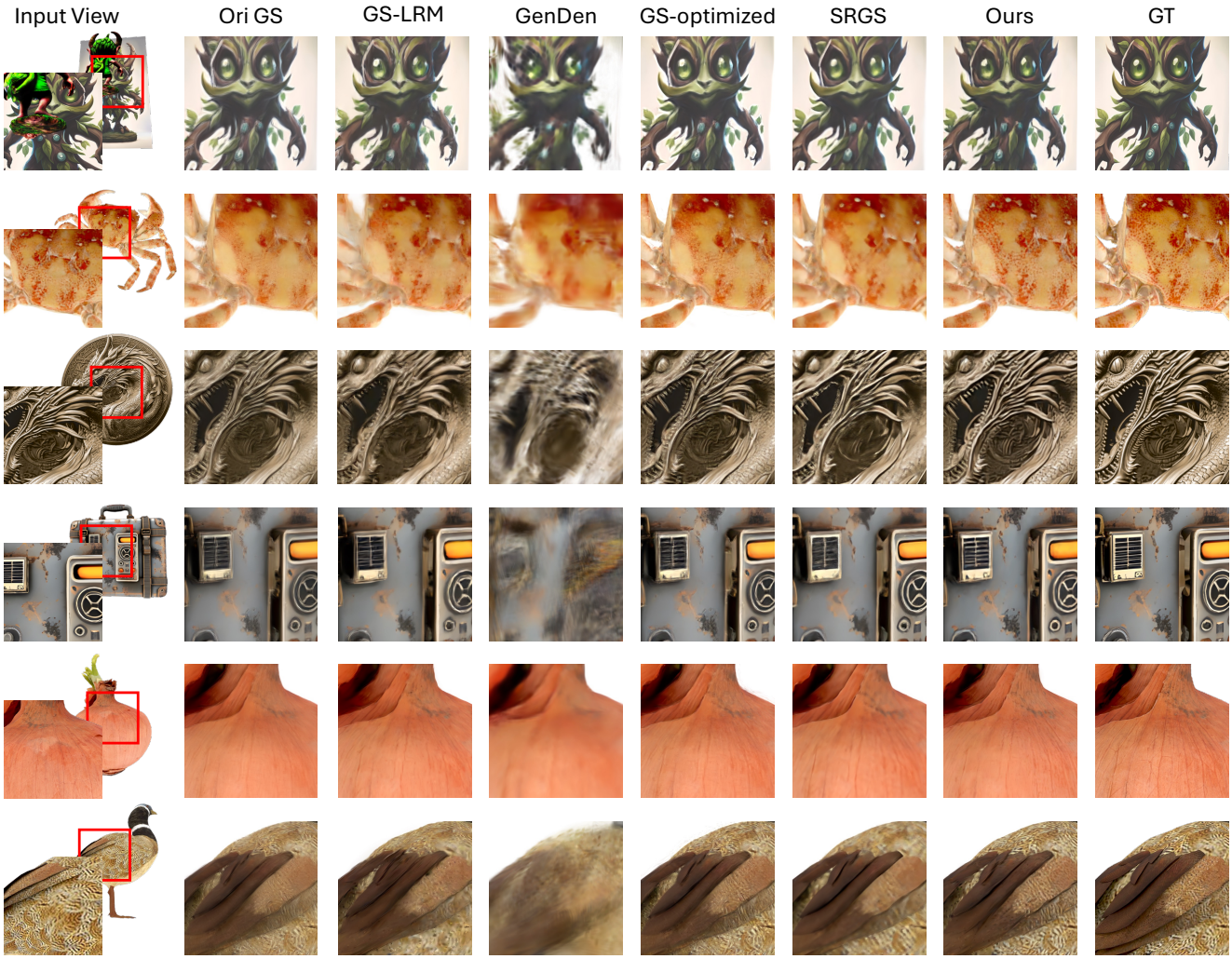

Figure 4. **Qualitative evaluation for local refinement**. Given a zoomed-in input view (left), we compare the refined results between baseline methods and ours from a novel view close to zoomed-in view. Our method recovers the details provided by the input zoomed-in views, and produces much sharper features compared to direct reconstruction (GS-LRM [94] or upsampling method (GenDen [50]) and is on-par or better than optimization-based approaches but using a fraction of the time (see Tab. 1).

Table 1. **Quantitative evaluation for local refinement.** Image quality is evaluated on two novel views close-by the provided zoomed-in images. Our method achieves the best performance on quality and efficiency, enabling interactive online refinement while preserving high-fidelity details.

Method	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Time (s)
GS-LRM [94]	20.28	0.5185	0.3873	0.92
GenDen [50]	17.32	0.5378	0.5979	1.68
3DGS [39]	20.52	0.6553	**0.3399**	323
SRGS [25]	21.01	**0.6759**	0.3540	495
Ours (*stage I*)	20.84	0.5698	0.3782	1.54
Ours (*stage II*)	**21.07**	0.5780	0.3673	1.54+0.64

and prior-based feedforward models (GenDen [50]), our approach better preserves high-frequency textures and fine-grained surface details. Optimization-based methods ([25, 39]) can improve local sharpness but require hundreds of seconds per update, making them impractical for interactive editing. In contrast, our model achieves comparable or higher quality. While stage I takes about 0.6 second, it needs to be done only once per asset, each subsequent edit takes about 0.3s (0.64s for two zoomed-in views as reported in Tab. 1), enabling interactive online applications.

4.3. Evaluation On Global Editing

Another important application of our framework is *global refinement*, for which we choose relighting as a representative task. In this setting, user provides one or more relit views and our model is tasked to propagate the tonal and

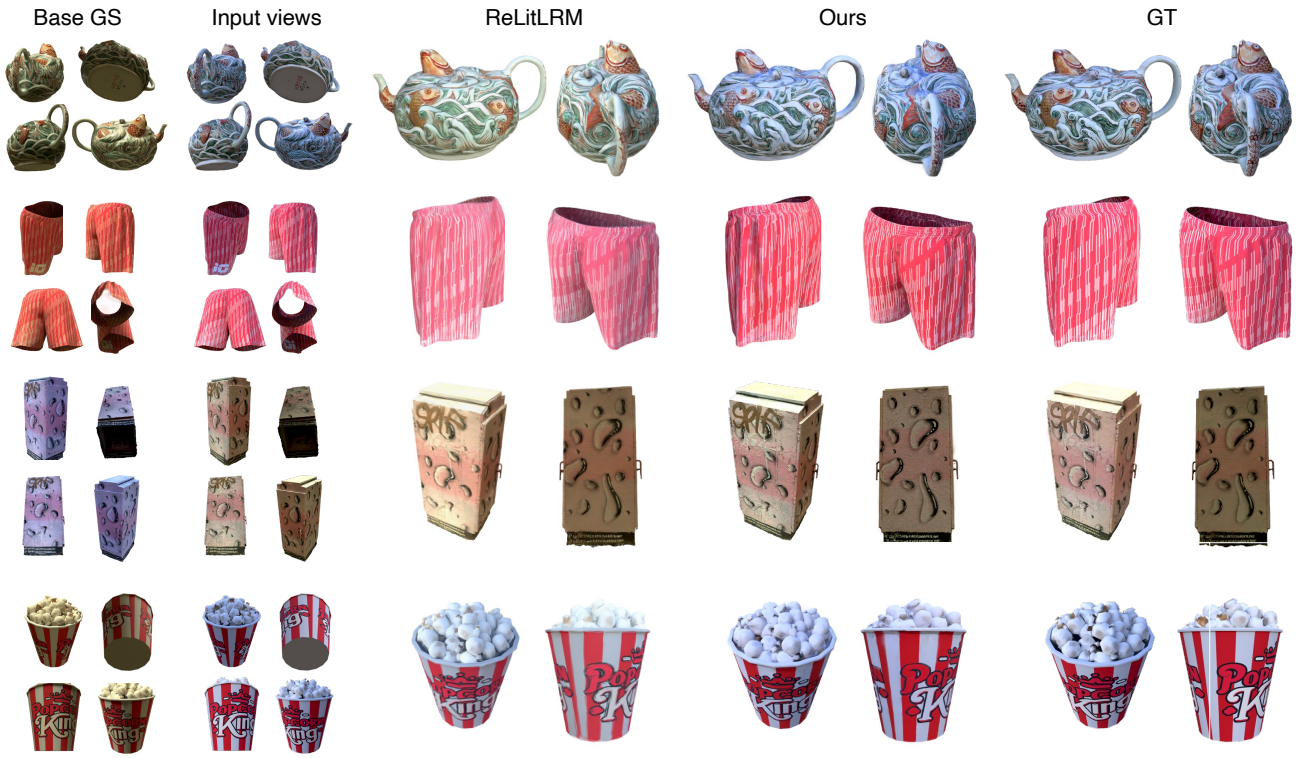

Figure 5. **Global relighting comparison**. ReLitLRM takes an environment map as input to synthesize new lighting, while our method leverages a few relit views as direct input. Though not strictly relighting, our method offers a practical path for appearance transfer, achieving consistent shadows, accurate color propagation, and closer alignment with ground truth.

shading changes to the entire 3D asset.

Baselines. While several works explore Gaussian editing via textual instructions [7, 30, 69], these approaches rely on text-conditioned diffusion priors and cannot incorporate explicit image-space modifications. While CMD [42] reports a related capability, the lack of released code and checkpoints prevents reproducible evaluation. Specifically, we compare against a generative relighting model, ReLitLRM [96], which takes the original multiview images and a target environment map as inputs, and synthesizes new spherical-harmonic coefficients via a diffusion model. Following the setup in Sec. 4.2, we use nine input views for Stage I and one to four relit views for Stage II. To ensure fair comparison, we align ReLitLRM's exposure levels with the provided relit views to mitigate lighting ambiguity.

Results. During testing, the refined Gaussians are rendered under eight novel camera views for evaluation. Tab. 2 demonstrates that our model consistently outperforms ReLitLRM across photometric metrics. Notably, our model operates at only $\sim$0.06 seconds per update (which is faster than refinement in local edits since no ray-tracing is needed for visible voxel selection). Qualitative comparisons in Fig. 5 show that our method achieves faithful global relighting with accurate color transfer, realistic shading, and strong view consistency. In contrast, ReLitLRM often exhibits mismatched color tones. In Fig. 7, we show the progression of the reconstruction accuracy given more relit input views. Interestingly, with only a single input view, our model produces visually coherent results in the unseen region, and reconstruction accuracy gradually increases with increasing input views (see the last three rows of Tab. 2).

Table 2. **Quantitative results for global relighting.** We evaluate novel-view reconstruction accuracy. Our method achieves the best quality while maintaining interactive speed. Performance improves steadily as more relit views are provided, demonstrating robust global propagation of lighting changes (also see Fig. 7).

Method	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Time (s)
RelitLRM [96]	21.68	0.8726	0.0912	22.1
Ours (*1 view*)	24.56	0.8986	0.0721	1.54+0.08
Ours (*2 views*)	24.71	0.8991	0.0719	1.54+0.11
Ours (*4 views*)	**25.03**	**0.9011**	**0.0712**	1.54+0.20

4.4. Ablation Study

GS-Voxel Compression. The goal of Stage I is to create a compact Gaussian representation while preserving visual fidelity. In Tab. 3, we ablate our voxel transformer design, using the uncompressed *Pre-Compression (GS-LRM)* output as a baseline to measure quality preservation. The re-

Table 3. **Evaluation of our voxel compression network**.To evaluate how well our method preserves visual quality, we compare different designs to the uncompressed GS-LRM baseline. Our chosen method (LocalAtt + feats query) achieves the highest reconstruction accuracy, demonstrating effective compression with minimal quality loss.

Method	PSNR (↑)	SSIM (↑)	LPIPS (↓)
VoxelAtt	30.65	0.9193	0.0804
LocalAtt + latent query	33.84	0.9349	0.0264
LocalAtt + feats query	**34.02**	**0.9387**	**0.0262**
Pre-Compression (GS-LRM)	34.19	0.9283	0.0249

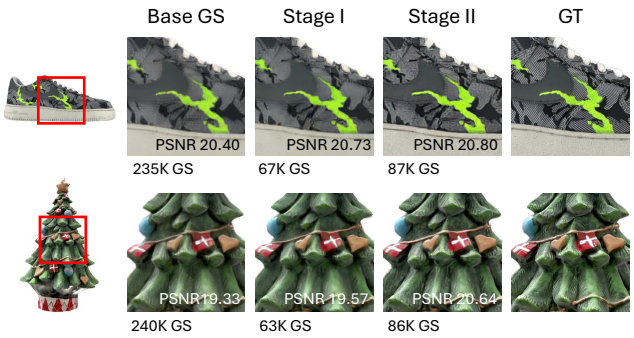

Figure 6. **The effect of our two-stage pipeline on a local refinement task.** Stage I achieves significant compression while maintaining visual fidelity. Stage II refines these compact GS, improving detail and overall quality based on the user's input.

sults show our final architecture (*LocalAtt + feats query*) is the most effective. We compare against two main variants. The first, *VoxelAtt*, degrades accuracy by averaging all Gaussian tokens in a voxel, which removes fine details. The second, *LocalAtt + latent query*, uses a single, shared latent embedding as a query for all voxels. This generic query is less effective than our approach, which is designed to preserve the most potent information from the original asset. Specifically, our method queries with features from the top-K opacity-weighted Gaussians within each voxel, allowing it to explicitly retain the most perceptually significant details and achieve superior reconstruction quality.

TTT Update. As shown in Fig. 7, our refinement quality steadily improves with additional input views. We further compare our TTT refinement with the cross-attention baseline by replacing TTT layers with cross attention layers in stage II in the global relighting task. The variant achieves 24.05 PSNR (-0.98 compared to ours) given four input relit views. While cross-attention enables feature fusion between the new input views and the existing Gaussian latents, it lacks adaptability once the parameters are fixed after training. In contrast, our TTT mechanism dynamically updates fast weights based on the input views, leading to more coherent color propagation and consistent illumination transfer across the entire scene. Please refer to our website for more results.

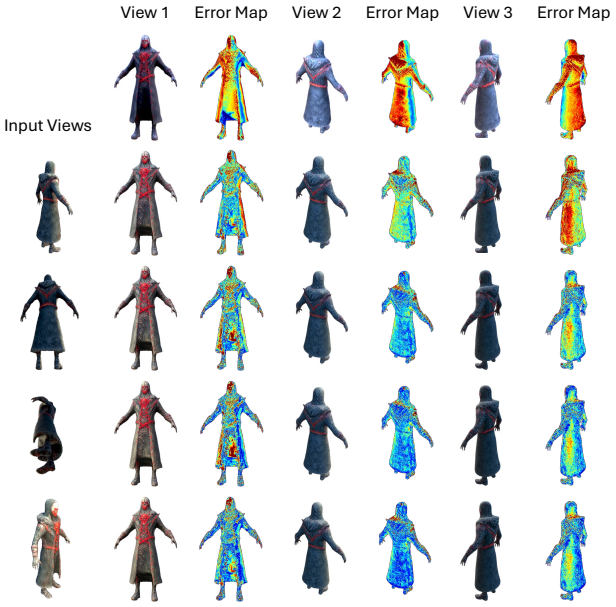

Figure 7. **Iterative updating on a global relighting task.** From top to bottom: original GS, refined GS with 1-4 input views. Our model picks up partial shading hint (see the first input view) and propagates it coherently to the unseen part (see view 1 in row 2).

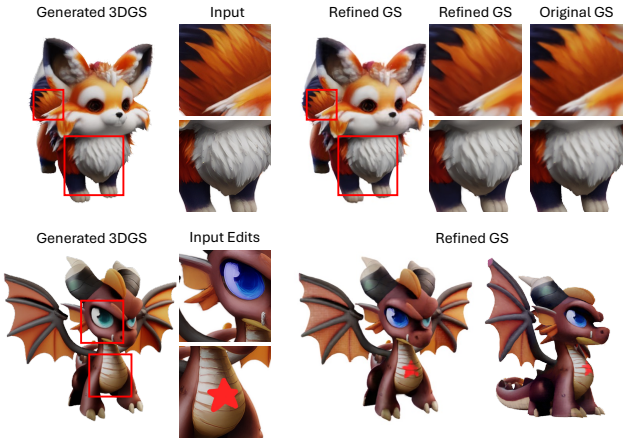

Figure 8. Results of our refinement model on generated 3DGS asset from Trellis [84]. Here we show refinement results from local super-resolution and editing (graffiti, recoloring) inputs (better viewed zoomed in).

Generalization. We further test our model on generated 3DGS from Trellis [84]. As shown in Fig. 8, our method generalizes well to the generated assets, effectively refining geometry-aligned textures from local super-resolution inputs and edits such as recoloring and graffiti.

5. Conclusion and Future Work

We presented SplatPainter, a feedforward, state-aware framework for interactive editing and refinement of exist-

ing 3D Gaussian Splatting (3DGS) assets. Our method introduces a compact voxel-based representation that substantially reduces redundancy while maintaining visual fidelity, and a test-time training (TTT) refinement module that efficiently incorporates new user inputs to achieve high-quality updates. Together, these designs enable both local and global editing tasks—such as super-resolution, graffiti, and relighting—in an interactive manner.

While our method focuses on appearance updates and does not introduce new geometry, future work could extend voxelization to a multiresolution design and incorporate geometric editing for scene-level manipulation.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011. 3

[2] Roi Bar-On, Dana Cohen-Bar, and Daniel Cohen-Or. Edit-p23: 3d editing via propagation of image prompts to multi-view. *arXiv preprint arXiv:2506.20652*, 2025. 2

[3] 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 *CVPR*, pages 16273–16282, 2025. 2

[4] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024. 4

[5] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv:2504.13173*, 2025. 4

[6] Hansheng Chen, Bokui Shen, Yulin Liu, Ruoxi Shi, Linqi Zhou, Connor Z Lin, Jiayuan Gu, Hao Su, Gordon Wetzstein, and Leonidas Guibas. 3d-adapter: Geometry-consistent multi-view diffusion for high-quality 3d generation. *arXiv preprint arXiv:2410.18974*, 2024. 2

[7] Minghao Chen, Iro Laina, and Andrea Vedaldi. Dge: Direct gaussian 3d editing by consistent multi-view editing. *arXiv preprint arXiv:2404.18929*, 2024. 2, 7

[8] 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. 2

[9] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Ttt3r: 3d reconstruction as test-time training. *arXiv preprint arXiv:2509.26645*, 2025. 3

[10] Yongwei Chen, Rui Chen, Jiabao Lei, Yabin Zhang, and Kui Jia. Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. *Advances in Neural Information Processing Systems*, 35:30923–30936, 2022. 2

[11] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21476–21485, 2024. 2

[12] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, pages 370–386. Springer, 2024. 3

[13] Yutong Chen, Marko Mihajlovic, Xiyi Chen, Yiming Wang, Sergey Prokudin, and Siyu Tang. Splatformer: Point transformer for robust 3d gaussian splatting, 2025. 2

[14] Yun-Chun Chen, Selena Ling, Zhiqin Chen, Vladimir G Kim, Matheus Gadelha, and Alec Jacobson. Text-guided controllable mesh refinement for interactive 3d modeling. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 2

[15] Yufeng Chi, Huimin Ma, Kafeng Wang, and Jianmin Li. Disco3d: Distilling multi-view consistency for 3d scene editing. *arXiv preprint arXiv:2508.01684*, 2025. 2

[16] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 3

[17] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. 5, 13

[18] 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. 2

[19] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023. 5, 13

[20] Nam Anh Dinh, Itai Lang, Hyunwoo Kim, Oded Stein, and Rana Hanocka. Geometry in style: 3d stylization via surface normal deformation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28456–28467, 2025. 2

[21] 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. 2

[22] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014. 3

[23] 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. 2

[24] Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3d gaussian splatting as new era: A

survey. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2

[25] Xiang Feng, Yongbo He, Yubo Wang, Yan Yang, Wen Li, Yifei Chen, Zhenzhong Kuang, Jianping Fan, Yu Jun, et al. Srgs: Super-resolution 3d gaussian splatting. *arXiv preprint arXiv:2404.10318*, 2024. 2, 5, 6

[26] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2016. 3

[27] Clement Fuji Tsang, Maria Shugrina, Jean Francois Lafleche, Or Perel, Charles Loop, Towaki Takikawa, Vismay Modi, Alexander Zook, Jiehan Wang, Wenzheng Chen, Tianchang Shen, Jun Gao, Krishna Murthy Jatavallabhula, Edward Smith, Artem Rozantsev, Sanja Fidler, Gavriel State, Jason Gorski, Tommy Xiang, Jianing Li, Michael Li, and Rev Lebaredian. Kaolin: A pytorch library for accelerating 3d deep learning research. 14

[28] William Gao, Noam Aigerman, Thibault Groueix, Vova Kim, and Rana Hanocka. Textdeformer: Geometry manipulation using text guidance. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–11, 2023. 2

[29] Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012. 3

[30] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 19740–19750, 2023. 2, 7

[31] Fangzhou Hong, Mingyuan Zhang, Liang Pan, Zhongang Cai, Lei Yang, and Ziwei Liu. Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. *arXiv preprint arXiv:2205.08535*, 2022. 2

[32] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023. 2, 5

[33] Jiajun Huang, Shuolin Xu, Hongchuan Yu, and Tong-Yee Lee. Gsdeformer: Direct, real-time and extensible cage-based deformation for 3d gaussian splatting, 2024. 2

[34] KIRI Innovations. Kiri engine: 3d scanner app for iphone, android, and web. https://www.kiriengine.com/, 2024. 2

[35] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021. 3

[36] Hiromichi Kamata, Yuiko Sakuma, Akio Hayakawa, Masato Ishii, and Takuya Narihira. Instruct 3d-to-3d: Text instruction guided 3d-to-3d conversion. *arXiv preprint arXiv:2303.15780*, 2023. 2

[37] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *Advances in neural information processing systems*, 30, 2017. 3

[38] Mahdi Karami and Vahab Mirrokni. Lattice: Learning to efficiently compress the memory. *arXiv preprint arXiv:2504.05646*, 2025. 4

[39] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1, 2, 5, 6, 13

[40] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *Advances in neural information processing systems*, 35:23311–23330, 2022. 2

[41] Jie Long Lee, Chen Li, and Gim Hee Lee. Disr-nerf: Diffusion-guided view-consistent super-resolution nerf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20561–20570, 2024. 2

[42] Peng Li, Suizhi Ma, Jialiang Chen, Yuan Liu, Congyi Zhang, Wei Xue, Wenhan Luo, Alla Sheffer, Wenping Wang, and Yike Guo. Cmd: Controllable multiview diffusion for 3d editing and progressive generation. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–10, 2025. 2, 7

[43] Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. Advances in 3d generation: A survey. *arXiv preprint arXiv:2401.17807*, 2024. 1

[44] Fangfu Liu, Hanyang Wang, Weiliang Chen, Haowen Sun, and Yueqi Duan. Make-your-3d: Fast and consistent subject-driven 3d content generation. In *European Conference on Computer Vision*, pages 389–406. Springer, 2024. 2

[45] Inc. Luma AI. Luma ai. https://lumalabs.ai/, 2024. 2

[46] Élie Michel and Adobe. Project new depth: Go from 2d to 3d in one click — adobe max sneaks. https://www.youtube.com/watch?v=1n_p3a-x3yE, 2023. Accessed: 2025-11-03. 2

[47] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13492–13502, 2022. 2

[48] 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

[49] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 3

[50] Seungtae Nam, Xiangyu Sun, Gyeongjin Kang, Younggeun Lee, Seungjun Oh, and Eunbyung Park. Generative densification: Learning to densify gaussians for high-fidelity generalizable 3d reconstruction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26683–26693, 2025. 2, 5, 6

[51] Jangho Park, Gihyun Kwon, and Jong Chul Ye. Ed-nerf: Efficient text-guided editing of 3d scene with latent space nerf. *arXiv preprint arXiv:2310.02712*, 2023. 2

[52] PlayCanvas. Supersplat: 3d gaussian splat editor. https://supersplat.com/, 2024. 2

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

[54] 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. 2

[55] 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 *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 2

[56] Nuri Ryu, Jiyun Won, Jooeun Son, Minsu Gong, Joo-Haeng Lee, and Sunghyun Cho. Elevating 3d models: High-quality texture and geometry refinement from a low-quality model. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–12, 2025. 2

[57] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021. 4

[58] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 3

[59] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997. 3

[60] 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. 2

[61] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. 4, 13

[62] Yuan Shen, Duygu Ceylan, Paul Guerrero, Zexiang Xu, Niloy J Mitra, Shenlong Wang, and Anna Frühstück. Supergaussian: Repurposing video models for 3d super resolution. In *European Conference on Computer Vision*, pages 215–233. Springer, 2024. 2

[63] Hyeonseop Song, Seokhun Choi, Hoseok Do, Chul Lee, and Taehyeong Kim. Blending-nerf: Text-driven localized editing in neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14383–14393, 2023. 2

[64] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15598–15607, 2021. 3

[65] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020. 2, 4

[66] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024. 4

[67] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021. 3

[68] Yifei Tong, Runze Tian, Xiao Han, Dingyao Liu, Fenggen Yu, and Yan Zhang. Cage-gs: High-fidelity cage based 3d gaussian splatting deformation. *arXiv preprint arXiv:2504.12800*, 2025. 2

[69] Cyrus Vachha and Ayaan Haque. Instruct-gs2gs: Editing 3d gaussian splats with instructions, 2024. 2, 7

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[71] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. In *European Conference on Computer Vision*, pages 439–457. Springer, 2024. 2

[72] Yecong Wan, Mingwen Shao, Yuanshuo Cheng, and Wangmeng Zuo. S2gaussian: Sparse-view super-resolution 3d gaussian splatting. *arXiv preprint arXiv:2503.04314*, 2025. 2

[73] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3835–3844, 2022. 2

[74] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Nerf-art: Text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics*, 30(8):4983–4996, 2023. 2

[75] Chen Wang, Hao-Yang Peng, Ying-Tian Liu, Jiatao Gu, and Shi-Min Hu. Diffusion models for 3d generation: A survey. *Computational Visual Media*, 11(1):1–28, 2025. 1

[76] Dongqing Wang, Tong Zhang, Alaa Abboud, and Sabine Süsstrunk. Inpaintnerf360: Text-guided 3d inpainting on unbounded neural radiance fields. *arXiv preprint arXiv:2305.15094*, 2(4):9, 2023. 2

[77] Hengyi Wang and Lourdes Agapito. 3d reconstruction with spatial memory. *arXiv preprint arXiv:2408.16061*, 2024. 3

[78] Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025. 2, 4

[79] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10510–10522, 2025. 3

[80] Minghao Wen, Shengjie Wu, Kangkan Wang, and Dong Liang. Intergsedit: Interactive 3d gaussian splatting editing with 3d geometry-consistent attention prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26136–26145, 2025. 2

[81] Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Prisacariu. GaussCtrl: Multi-View Consistent Text-Driven 3D Gaussian Splatting Editing. *ECCV*, 2024. 2

[82] Jiatong Xia and Lingqiao Liu. Close-up-gs: Enhancing close-up view synthesis in 3d gaussian splatting with progressive self-training. *arXiv preprint arXiv:2503.09396*, 2025. 2

[83] Jiatong Xia, Libo Sun, and Lingqiao Liu. Enhancing close-up novel view synthesis via pseudo-labeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8567–8574, 2025. 2

[84] 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. *arXiv preprint arXiv:2412.01506*, 2024. 8

[85] Shiyun Xie, Zhiru Wang, Yinghao Zhu, and Chengwei Pan. Supergs: Super-resolution 3d gaussian splatting via latent feature field and gradient-guided splitting. *arXiv preprint arXiv:2410.02571*, 2024. 2

[86] Tianhao Xie, Noam Aigerman, Eugene Belilovsky, and Tiberiu Popa. Sketch-guided cage-based 3d gaussian splatting deformation. *arXiv preprint arXiv:2411.12168*, 2024. 2

[87] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024. 2

[88] Teng Xu, Jiamin Chen, Peng Chen, Youjia Zhang, Junqing Yu, and Wei Yang. Tiger: Text-instructed 3d gaussian retrieval and coherent editing. *arXiv preprint arXiv:2405.14455*, 2024. 2

[89] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. In *European Conference on Computer Vision*, pages 1–20. Springer, 2024. 2, 3

[90] Youngho Yoon and Kuk-Jin Yoon. Cross-guided optimization of radiance fields with multi-view image super-resolution for high-resolution novel view synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12428–12438, 2023. 2

[91] Xiqian Yu, Hanxin Zhu, Tianyu He, and Zhibo Chen. Gaussiansr: 3d gaussian super-resolution with 2d diffusion priors. *arXiv preprint arXiv:2406.10111*, 2024. 2

[92] Lin Zeng, Boming Zhao, Jiarui Hu, Xujie Shen, Ziqiang Dang, Hujun Bao, and Zhaopeng Cui. Gaussianupdate: Continual 3d gaussian splatting update for changing environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25800–25809, 2025. 3

[93] Hao Zhang, Yao Feng, Peter Kulits, Yandong Wen, Justus Thies, and Michael J Black. Text-guided generation and editing of compositional 3d avatars. *arXiv preprint arXiv:2309.07125*, 2023. 2

[94] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *European Conference on Computer Vision*, 2024. 2, 3, 5, 6, 13

[95] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[96] Tianyuan Zhang, Zhengfei Kuang, Haian Jin, Zexiang Xu, Sai Bi, Hao Tan, He Zhang, Yiwei Hu, Milos Hasan, William T Freeman, et al. Relitlrm: Generative relightable radiance for large reconstruction models. *arXiv preprint arXiv:2410.06231*, 2024. 7

[97] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025. 2, 3, 4, 5, 13

[98] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5449–5458, 2022. 3

[99] Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. Texverse: A universe of 3d objects with high-resolution textures. *arXiv preprint arXiv:2508.10868*, 2025. 5

[100] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024. 2

[101] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *2024 International Conference on 3D Vision (3DV)*, pages 42–52. IEEE, 2024. 3

[102] Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–10, 2023. 2

[103] 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. 2

Supplementary Materials

A. Implementation Details

In this section, we provide additional details on the implementation of our method.

A.1. Network Implementation and Training

Backbone and Feature Extraction. We adopt GS-LRM [94] as our Gaussian large reconstruction model (LRM) backbone. Given input multiview images (either coming from reconstruction inputs or rendered from a pre-generated asset), we first reconstruct a 3D Gaussian scene and its associated image tokens. The image tokenizer and transformer backbone in GS-LRM are kept frozen in all of our experiments; we only use them to provide feature-rich tokens for our two-stage network.

Stage I: Local Voxel Transformer. Stage I compresses the dense GS-LRM output into compact voxel latents. We voxelize the reconstructed 3DGS using voxel resolution as 128^3 and group Gaussians into voxels as described in **Sec. 3.1**. Each Gaussian is encoded by concatenating its GS-LRM feature token and Gaussian attributes (position, scale, opacity, rotation, spherical harmonics), followed by a linear projection to a 512 dimensional embedding.

For each voxel, we apply a transformer encoder with $L_{\text{enc}} = 6$ layers. Each layer consists of multi-head self-attention (8 heads, hidden size 512) and a feedforward MLP with SwiGLU [61] activation, both with residual connections and layer normalization. We denote the encoded per-Gaussian latents as $\{z_k\}_{k=1}^{N_k}$. To aggregate information, we select the top-K Gaussians in each voxel based on opacity ($K = \max(\lfloor 0.25 N_k \rfloor, 1)$) and use their latents as queries in a cross-attention layer. A transformer decoder with $L_{\text{dec}} = 6$ self-attention layers then produces K compact latent per voxel, $\tilde{\mathcal{V}}$, which we store as our compressed GS representation. Opacity-weighted queries focus on visible, perceptually dominant regions and empirically yield higher reconstruction quality (see **Tab. 3** in the main paper). For efficient batch training, all voxels are flattened and concatenated into a single 1D sequence, with per-voxel sequence lengths tracked explicitly. We use FlashAttention2 [17] in packed-attention mode to support variable-length voxel groups without padding overhead.

Stage II: TTT Refinement Network. Stage II refines voxel latents using new user-provided views. The refinement module is a stack of $L_{\text{ttt}} = 8$ TTT blocks (**Sec. 3.2**), each consisting of: (i) a large-chunk TTT layer [97] operating on image tokens and voxel latents, followed by (ii) a feedforward MLP with SwiGLU [61] activation. The mapping network f_W in each TTT layer is implemented as a 2-layer SwiGLU-MLP [61] with hidden size 2048 and output dimension 512. Fast weights W are maintained per block, and only W is updated during test-time training; all slow weights (projection heads, MLPs, and GS-LRM) remain frozen.

GS Decoder and Renderer. To map voxel latents back to Gaussians, we use a lightweight decoder consisting of one linear layer. The decoder predicts Gaussian attributes (position, scale, opacity, rotation, SH coefficients). We render the updated Gaussians using the original 3DGS renderer [39] to produce I_{render}.

Training Objectives. Both Stage I and Stage II are trained with an image-based reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \|I_{\text{render}} - I_{\text{gt}}\|_2 + \lambda_{\text{perc}} \big\|\phi(I_{\text{render}}) - \phi(I_{\text{gt}})\big\|_2, \quad (9)$$

as introduced in **Sec. 3.3**. We set $\lambda_{\text{perc}} = 0.5$ in all experiments. For Stage I, I_{render} uses randomly selected views. For Stage II, I_{render} corresponds to: (i) zoomed-in crops for local refinement plus randomly selected views showing the whole objects, or (ii) globally relit views for the relighting task. We train Stage I and Stage II sequentially. Stage I is optimized for 80k iterations using AdamW with learning rate 4×10^{-5}, batch size 1, and cosine learning-rate decay. We then freeze Stage I (including the GS decoder) and train Stage II for another 200k iterations with AdamW, learning rate 4×10^{-5}, batch size 1, and the same cosine schedule. Stage I is trained using Objaverse [19] data, and we have two Stage II models trained on different data based on applications (local edits and global relighting). We aim at future work for training a unified model to support various applications. All experiments are conducted on 16×A100 80GB GPUs with mixed-precision training.

A.2. Inference

Preprocessing (One-Time per Asset). Given a new 3D Gaussian asset, we first run GS-LRM [94] to obtain dense Gaussians and image tokens. We then voxelize the scene and run Stage I to obtain compact voxel latents $\tilde{\mathcal{V}}$ and the corresponding decoder that maps them back to Gaussians. This preprocessing step is executed once per asset and takes $\sim$1.5 s for the resolutions used in our experiments.

Interactive Editing Loop. At inference time, user edits are provided as a small number of 2D images:

- **Local refinement.** The user supplies one or more high-resolution zoom-in views, optionally with paint-over graffiti. We associate each edited view with the visible

voxels via ray tracing implemented based on Kaolin Library [27] and only update latents in those voxels. The image tokens of new input views are fed into Stage II, where TTT layers update fast weights W using these image tokens, and the updated W is then applied to the corresponding selected visible voxel latents. We decode updated Gaussians and re-render the scene. A single local edit takes $\sim$0.3 s (including ray-voxel association and rendering).

- **Global relighting.** The user provides one to four relit views under a new HDRI. In this case, all voxel latents are refined by Stage II without voxel-level masking or ray tracing. The updated Gaussians are rendered under novel views to visualize the new lighting. Each global update takes $\sim$0.06 s, enabling interactive relighting.

B. Additional Results

Please refer to our website for additional visual results.