Velocity-Space 3D Asset Editing

Hao Liu[1,3] **Yuxuan Lin**[1] **Jingfeng Guo**[2] **Ruihang Chu**[1] **Junjie Wang**[1]
Ruotong Li[3†] **Yujiu Yang**[1†]

[1]Tsinghua University [2]South China University of Technology [3]Pengcheng Laboratory

{h-l25, rhchu}@mails.tsinghua.edu.cn, linyx23@tsinghua.org.cn
drjfguo@gmail.com, {wangjunjie, yang.yujiu}@sz.tsinghua.edu.cn, lirt_cs@163.com

Abstract

Editing a 3D asset locally, modifying a target region while preserving the rest, is a fundamental requirement of native 3D editing. Existing methods enforce locality through mechanisms external to the generator, such as manual 3D masks, post-hoc voxel merging, or 2D multi-view lifting. None of them intervene where the corruption actually originates: inside the ODE sampler. For a rectified-flow generator to achieve faithful local editing, its velocity field should be strong over the target editing region while vanishing on preserved content. Yet a single velocity field can hardly satisfy both requirements simultaneously, leading to three problems: (i) identity leakage that keeps the edit signal non-zero on preserved regions; (ii) no dedicated edit-amplification channel, so strengthening the edit inevitably perturbs identity; and (iii) an identity drag at the geometry and material stages, where a global condition pulls every token toward the target. We propose **VS3D** (*Velocity-**S**pace **3D** Asset editing*), an **inversion-free, training-free, and mask-free** framework that derives edit and preserve signals by comparing the frozen model's source- and target-conditioned responses and realizes this principle through three stage-specific modules. Reconstruction-Anchored Source Injection (RASI) absorbs identity leakage by turning the unconditional embedding into a per-step, asset-specific anchor calibrated through source reconstruction. Partial-Mean Guidance (PMG) amplifies the edit signal by contrasting high- and low-quality subsample estimates of the velocity difference, active only where a consistent edit exists. Twin-Agreement Residual Injection (TAR) lets the sampler decide token by token what to preserve at the geometry and material stages. Experiments on diverse 3D assets show that VS3D outperforms state-of-the-art native-3D and 2D-lifted editors, demonstrating the effectiveness of velocity-space editing for training-free 3D asset editing. Code is publicly available at `https://github.com/LHaoooo/VS3D`.

1 Introduction

Precise local editing of 3D assets is essential for content creation, game development, and embodied simulation, yet it poses a fundamental dual requirement on any pretrained generator: the model must apply a strong, targeted modification to the region the user specifies while leaving every other region strictly unchanged. The dominant family of native 3D generators [7, 17–20] now builds on rectified-flow DiTs [4, 3], where generation is an ODE whose velocity field is learned by conditional flow matching. Editing with such a model therefore reduces to *controlling its velocity field*: the velocity update must carry a non-trivial edit signal on the target region and vanish on the rest. How to achieve this control on a frozen 3D rectified-flow DiT, without resorting to external masks or additional training, remains an open problem.

†Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Existing 3D editors typically enforce source preservation through mechanisms outside the velocity-field update. Three routes recur: (i) *mask-based editors* [8–10, 12] ask the user or an external tool to supply a binary 3D region that pins every non-edited voxel to the source, tedious to draw on a voxel grid, brittle for natural edits whose support fades gradually, and prone to seam artefacts wherever a hard binary boundary meets smooth geometry; (ii) *post-hoc mergers* [14] try to recover identity by connected-component stitching that can neither distinguish genuine edits from leakage nor suppress boundary artefacts at the merge seam; and (iii) *multi-view / video editors* [15, 11, 16] enforce identity in 2D and lift the result back through a reconstructor, yet edits that alter topology or volumetric occupancy introduce multi-view inconsistencies that accumulate across views. All three share a deeper limitation: because **the velocity field of a rectified-flow ODE is the sole carrier of both identity and edit signals**, the most direct path to simultaneous identity preservation and edit amplification is to control that velocity field from within; none of the existing routes attempt this, and instead impose identity from outside the ODE without ever examining the ODE itself.

Among inversion-free velocity-space editing formulations, FlowEdit [2] couples a source and a target branch through shared noise so that their differential velocity v_Δ directly drives the edit offset without inverting the source into noise. We adopt this coupling and *examine the ODE itself* to understand why identity still drifts. Our analysis reveals three problems. **(P1) Two-channel identity leakage.** Even when the two conditions locally agree, v_Δ does not vanish on preserved content: it carries a *CFG-asymmetry residual*, structurally non-zero because effective editing requires a much stronger target guidance than source guidance, and an *attention-mediated condition residual*, because the conditioning features are a sequence of tokens extracted from a single image, whose perturbations can propagate through cross-attention to voxel tokens outside the edited region. Euler integration accumulates both residuals and visibly deforms non-edited regions (closed-form derivation in §3.2). **(P2) Absence of a dedicated edit-amplification channel.** Identity preservation and edit amplification compete for the same degrees of freedom within the velocity field. Because few-sample inference is unavoidable for costly 3D DiTs, the Monte-Carlo estimate of v_Δ is dominated by estimation variance rather than the consistent edit signal, leaving the edit region under-expressed even when the leakage has been suppressed. **(P3) Global-condition override.** The downstream geometry and material stages refine shape and appearance on the coordinate occupancy produced by the sparse-occupancy stage, each conditioned on feature tokens extracted from a single *target* image, which can pull non-edited tokens toward the target appearance. Without a counterbalancing mechanism, this global drag overwrites the source identity on preserved tokens, causing geometry distortion and texture drift. VS3D addresses these failures inside the sampler at the stage where each problem arises.

We address each problem with a targeted intervention inside the ODE, and the three solutions share a common insight: edit-sensitive content is identified by disagreement between the frozen model's source- and target-conditioned responses, while agreement indicates content to preserve. Because this signal appears differently in dense occupancy latents and sparse geometry/material SLATs, VS3D realizes the same criterion with stage-specific mechanisms. For **P1**, Reconstruction-Anchored Source Injection (RASI) turns the unconditional embedding into a per-step, asset-specific handle optimised through a source-reconstruction probe, absorbing both leakage channels at once so that v_Δ collapses toward zero on non-edited voxels without any external region indication. For **P2**, Partial-Mean Guidance (PMG) contrasts a high-quality and a low-quality subsample estimate of the edit direction and extrapolates away from the noisier one; the amplification is automatically gated by v_Δ itself, active only where a consistent edit signal survives and inert where RASI has already suppressed the velocity. For **P3**, Twin-Agreement Residual Injection (TAR) runs a second sparse-DiT pass on the same coordinate occupancy, but conditioned on the source image, and measures per-token agreement; tokens on which the two passes agree are blended back toward the source encoding, while the edit region is left untouched. We refer to the resulting system as **VS3D**, an inversion-free, training-free, mask-free native-3D editor that achieves high-fidelity local edits with strong identity preservation entirely inside the ODE, without relying on external mechanisms to enforce edit locality.

Contributions. Our main contributions are as follows:

- We formulate local 3D editing as a model-internal edit/preserve problem in velocity space, where disagreement between source- and target-conditioned responses indicates edit-sensitive content and agreement indicates content to preserve.

- We implement this criterion with stage-specific modules: Reconstruction-Anchored Source Injection (RASI; §3.2) and Partial-Mean Guidance (PMG; §3.3) act on dense occupancy latents to suppress leakage and amplify edits, while Twin-Agreement Residual Injection (TAR; §3.4) acts on sparse geometry and material SLATs to preserve non-edited tokens.
- We build **VS3D**, an inversion-free, training-free, and mask-free full-asset editing framework that edits occupancy, geometry, and material within a frozen 3D ODE sampler.

2 Background

Native structured-latent 3D generator. We adopt TRELLIS 2.0 [7] as a frozen backbone: a native 3D generator whose generation process unfolds in three stages with three independent DiTs, all conditioned on a DINOv3-L [5] embedding of the input image. The three stages are (i) *sparse-structure generation*, (ii) *geometry generation*, and (iii) *material generation*. Stage 1 operates on a *dense* low-resolution latent $z^{\mathrm{ss}} \in \mathbb{R}^{C\times R\times R\times R}$ (R=64 in our 1024^3-output setting) and is decoded into a binary occupancy grid whose active coordinates $\mathcal{V} \subset \mathbb{Z}^3$ define the voxel support of the subsequent stages. Stages 2–3 use sparse DiTs on $\mathcal{V}$, with separate structured latents (SLATs) for geometry and material. Two Sparse Compression VAEs (SC-VAEs), each with $16\times$ spatial compression, decode these latents into geometry and PBR material fields (albedo, roughness, metallic, and normal). During upsampling, the material SC-VAE is conditioned on the geometry decoder's subdivision structure [7], aligning the geometry and PBR fields on the shared O-Voxel support.

Rectified flow. Each stage of TRELLIS 2.0 is a rectified flow [4] trained under the conditional flow-matching objective [3]. For a clean latent x_0 and Gaussian noise $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$, the forward process is the straight-line interpolation

$$x(t) \;=\; (1-t)\, x_0 \;+\; t\,\varepsilon, \qquad t \in [0,1], \tag{1}$$

whose constant time derivative $\varepsilon - x_0$ is the regression target of the velocity network $v_\theta(x, t, c)$, where c denotes the image-conditioning signal. Setting $v_\theta(x, t, c) \approx \varepsilon - x_0$ and eliminating ε via (1) gives the implicit clean-sample estimate

$$\hat{x}_0(x, t, c) \;=\; x - t\, v_\theta(x, t, c). \tag{2}$$

Inversion-free velocity-space editing. Given the clean source latent x_{src} and conditions $c_{\mathrm{src}}, c_{\mathrm{tgt}}$, FlowEdit [2] transports x_{src} to an edited latent without ever inverting it into a noise latent. It maintains an edit state z_t^{edit} initialised at $z_1^{\mathrm{edit}}{=}x_{\mathrm{src}}$ and integrated from $t{=}1$ down to $t{=}0$ along a discrete schedule $1{=}t_0{>}t_1{>}\cdots{>}t_T{=}0$ with step size $\Delta t{=}t_{k+1} - t_k$. At each step the two branches are coupled by a shared noise draw $\varepsilon{\sim}\mathcal{N}(0, \mathbf{I})$ on the interpolation of (1),

$$z_t^{\mathrm{src}} = (1-t)\, x_{\mathrm{src}} + t\,\varepsilon, \qquad z_t^{\mathrm{tgt}} = z_t^{\mathrm{edit}} + \big(z_t^{\mathrm{src}} - x_{\mathrm{src}}\big), \tag{3}$$

so that $z_t^{\mathrm{tgt}} - z_t^{\mathrm{src}}$ equals the running edit offset $z_t^{\mathrm{edit}} - x_{\mathrm{src}}$ at every t. Each branch is then sampled under its own condition and its own classifier-free guidance weight,

$$\tilde{v}_\theta(z_t, t, c, \phi; \omega) \;=\; (1+\omega)\, v_\theta(z_t, t, c) \;-\; \omega\, v_\theta(z_t, t, \phi), \tag{4}$$

where ω is the guidance weight and ϕ is an unconditional embedding (by default the network's built-in null ϕ_0). Since $z_t^{\mathrm{tgt}} - z_t^{\mathrm{src}}$ tracks the edit offset itself, the direction along which this offset should evolve is exactly the difference of the two branches' guided velocities:

$$v_\Delta(z_t^{\mathrm{edit}}, t) \;=\; \tilde{v}_\theta(z_t^{\mathrm{tgt}}, t, c_{\mathrm{tgt}}, \phi; \omega_{\mathrm{tgt}}) \;-\; \tilde{v}_\theta(z_t^{\mathrm{src}}, t, c_{\mathrm{src}}, \phi; \omega_{\mathrm{src}}). \tag{5}$$

The edit state is advanced by one Euler step $z_{t_{k+1}}^{\mathrm{edit}} = z_{t_k}^{\mathrm{edit}} + \Delta t\, v_\Delta(z_{t_k}^{\mathrm{edit}}, t_k)$, where v_Δ is estimated as a **Monte-Carlo average over S independent noise realisations** of the coupling (3), with S kept small on a 3D rectified-flow DiT due to per-step cost. Integrating v_Δ thus transports x_{src} to an edited latent directly in velocity space, without ever inverting the source into a noise latent.

3 VS3D

3.1 Overview

VS3D addresses mask-free local editing with three modules acting on the ODE, each targeting a distinct failure mode (Fig. 1). The editing problem naturally splits into two regimes. The *dense* Stage-1

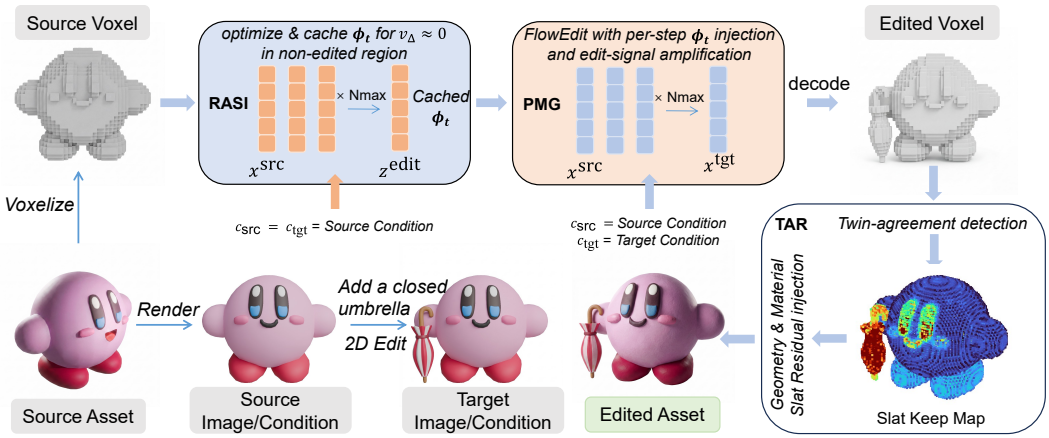


Figure 1: **Overview of the VS3D pipeline.** The source 3D asset is rendered to obtain a source image, which is edited in 2D to produce the target condition image. Stage 1 operates on the dense occupancy latent: *RASI* (§3.2) optimises a per-step ϕ_t to suppress v_Δ on non-edited regions, and *PMG* (§3.3) amplifies the edit signal via subsample extrapolation. Stages 2–3 handle sparse geometry and material SLATs: *TAR* (§3.4) computes a token-wise p_{flow} map (blue = preserve, red = edit) and injects source residuals accordingly to produce the final edited asset.

occupancy latent admits FlowEdit's coupling (3), so editing reduces to shaping v_Δ; however, two leakage channels—CFG asymmetry and the global-condition gap (§3.2)—keep v_Δ structurally non-zero on preserved content. The *sparse* Stage-2/3 geometry and material SLATs have mismatched active coordinates between source and target, making velocity-space coupling unavailable; DINOv3 feature tokens extracted from a single image can spread changes in the target condition to non-edited tokens, so the challenge becomes *which tokens to leave alone* (§3.4).

Our three modules resolve these issues in sequence: **(i)** *Reconstruction-Anchored Source Injection* (RASI, §3.2) closes Stage-1 identity leakage by turning the null embedding into a per-step, asset-specific ϕ_{t_k} fitted via a source-reconstruction probe, absorbing both leakage channels simultaneously. **(ii)** *Partial-Mean Guidance* (PMG, §3.3) amplifies the edit signal by extrapolating from a noisier subsample mean of v_Δ toward a cleaner one; the amplification is automatically gated by v_Δ itself, active only where a consistent edit exists. **(iii)** *Twin-Agreement Residual Injection* (TAR, §3.4) lets the Stage-2/3 sparse-DiT sampler decide token by token what to preserve and what to regenerate, handling additions, removals, and replacements uniformly.

Problem formulation. Given a source 3D asset $\mathcal{A}_{\text{src}}$ with geometry and material, and a pair of reference images $(I_{\text{src}}, I_{\text{tgt}})$ depicting its current and desired appearance, we seek an edited asset that realises the edit in the implied region while leaving the rest intact, under two hard constraints:

- **Training-free**: no model or LoRA is fine-tuned; all parameters of v_θ are frozen.
- **Mask-free**: neither manual masks nor external segmentation networks are used to delineate the edit region.

3.2 Reconstruction-Anchored Source Injection for Identity-Leakage Closure

The velocity difference v_Δ should vanish on non-edited voxels, yet off-the-shelf FlowEdit leaves a persistent residual that Euler integration accumulates into identity drift. We trace this residual to two structural channels and close both by turning the unconditional embedding ϕ into a per-step, asset-specific calibration variable optimised through a source-reconstruction probe.

Problem diagnosis. To isolate the first leakage channel, consider forcing $c_{\text{tgt}} = c_{\text{src}}$: the two branches of (3) then share the same condition and should ideally give $v_\Delta = 0$, yet substituting $c_{\text{tgt}} = c_{\text{src}}$ into (5) and expanding (4) yields

$$v_\Delta\big|_{c_{\text{tgt}}=c_{\text{src}}} = (\omega_{\text{tgt}} - \omega_{\text{src}}) \left[v_\theta(z_t^{\text{src}}, t, c_{\text{src}}) - v_\theta(z_t^{\text{src}}, t, \phi) \right]. \tag{6}$$

Effective editing adopts $\omega_{\text{tgt}} \gg \omega_{\text{src}}$ for the target condition to take effect, so the prefactor in (6) is structurally non-zero even in this degenerate regime: asymmetric CFG weights alone are already enough to keep v_Δ away from zero, independent of any condition gap. Second, local agreement on a non-edited region does not imply global agreement: since DINOv3 encodes each image into a sequence of contextualized feature tokens, any difference in the edited region perturbs $c_{\text{src}}, c_{\text{tgt}}$ as a whole, and dense cross-attention then propagates this gap into every voxel token, so $v_\theta(\cdot, t, c_{\text{src}}) \neq v_\theta(\cdot, t, c_{\text{tgt}})$ even on non-edited voxels. Euler integration accumulates both residuals into z_t^{edit} along the non-edited support, and mask-free editing cannot remove them after the fact.

Absorbing leakage via ϕ. Among the symbols in (6), v_θ is frozen, $c_{\text{src}}, c_{\text{tgt}}$ carry the edit semantics, and $\omega_{\text{tgt}}, \omega_{\text{src}}$ set the edit strength. **The unconditional embedding ϕ is the natural absorption site for both leakage channels**: optimising ϕ per timestep encodes asset-specific reconstruction information without interfering with the condition or the guidance schedule [1]. We design a source-reconstruction probe: at each active step t_k, we replace c_{tgt} with c_{src} on the target branch while keeping its guidance weight at ω_{tgt}, and ask whether a single Euler step of v_Δ can bring $z_{t_k}^{\text{edit}}$ back to x_{src}. Switching only the condition but not the CFG weight reproduces the exact guidance schedule used at editing time, so both leakage channels, including the $(\omega_{\text{tgt}} - \omega_{\text{src}})$ prefactor, are present in the probe and available to be absorbed by ϕ. Concretely, we solve

$$\phi_{t_k} \;=\; \arg\min_{\phi} \left\| z_{t_k}^{\text{edit}} + \Delta t\big(\tilde{v}_\theta(z_{t_k}^{\text{tgt}}, t_k, c_{\text{src}}, \phi; \omega_{\text{tgt}}) - \tilde{v}_\theta(z_{t_k}^{\text{src}}, t_k, c_{\text{src}}, \phi; \omega_{\text{src}})\big) \;-\; x_{\text{src}} \right\|_2^2, \quad (7)$$

where $z_{t_k}^{\text{src}}, z_{t_k}^{\text{tgt}}$ are formed via the coupling (3), *both* branches are conditioned on c_{src}, and each branch retains its real-editing guidance weight. Minimising the deviation from x_{src} forces ϕ_{t_k} to absorb both leakage channels into a single correction tied to this asset and this timestep. We then advance z^{edit} one Euler step along the probe ODE and cache ϕ_{t_k}; optimisation hyper-parameters are reported in §A.1.

During editing (§3.3), we evaluate v_Δ with c_{tgt} on the target branch and c_{src} on the source branch, substituting the cached ϕ_t for ϕ_0 in both CFG calls. Because the probe has pinned the c_{src}-only behaviour of v_Δ to self-reconstruction, the real update retains only the c_{tgt}-driven edit signal: on non-edited voxels v_Δ collapses toward zero without any mask, turning identity preservation into a property of the velocity field itself.

3.3 Partial-Mean Guidance for Edit-Signal Amplification

RASI secures identity on non-edited voxels, but the same optimisation may inadvertently suppress v_Δ on edited ones as well, weakening the edit signal. We need an amplification handle that is automatically gated by the edit region—active where v_Δ is large, silent where RASI has driven it toward zero—so that edit strength and identity preservation are decoupled without any mask. We obtain such a handle from the Monte-Carlo structure of v_Δ itself, at no extra forward cost, by extrapolating between a full-sample mean and a noisier partial-sample mean.

Why a separate amplifier is needed. Because the RASI objective (7) asks one Euler step of v_Δ to reconstruct x_{src}, the optimised ϕ_t may absorb v_Δ on edited voxels too, leaving the surviving edit signal as whatever the optimiser could not capture.

Extrapolation between subsample means. Recall that v_Δ (5) is estimated via the shared noise ε in the coupling (3). Each independent draw $\varepsilon^{(s)}$ produces a velocity difference $v_\Delta^{(s)}$, which we decompose as $v_\Delta^{(s)} = \mu + \eta^{(s)}$ where μ is the conditional expectation (the true edit signal) and $\eta^{(s)}$ is a zero-mean noise term. The S-sample mean $\hat{\mu}_S = \frac{1}{S}\sum_{s=1}^{S} v_\Delta^{(s)}$ and any partial mean $\hat{\mu}_L = \frac{1}{L}\sum_{s=1}^{L} v_\Delta^{(s)}$ with $1 \leq L < S$ are both unbiased for μ but with variances σ_η^2/S and σ_η^2/L respectively. Extrapolating away from the noisier estimate amplifies the direction in which the full-sample estimate is more confident:

$$\hat{\mu}_S \;+\; w\,(\hat{\mu}_S - \hat{\mu}_L) \;=\; (1+w)\,\hat{\mu}_S \;-\; w\,\hat{\mu}_L, \qquad w \in \mathbb{R}_{\geq 0}. \quad (8)$$

The gated behaviour emerges automatically. On non-edited voxels RASI has driven μ toward zero, so $\hat{\mu}_S$ and $\hat{\mu}_L$ both hover near zero and their gap has no consistent direction; the extrapolation

is negligible and identity is not perturbed. On edited voxels μ is large, $\hat{\mu}_S$ tracks it more faithfully than $\hat{\mu}_L$, and (8) lengthens exactly the edit-aligned component. The full Stage-1 update at active step t_k becomes

$$z^{\text{edit}}_{t_{k+1}} \;=\; z^{\text{edit}}_{t_k} \;+\; \Delta t\,\big[\,\hat{\mu}_S \;+\; w\,(\hat{\mu}_S - \hat{\mu}_L)\,\big], \tag{9}$$

where every $v^{(s)}_{\Delta}$ already carries the injected ϕ_t from RASI.

3.4 Twin-Agreement Residual Injection for Sparse-Stage Preservation

Stage 1 determines the edited coordinates $\mathcal{C}_{\text{tgt}}$ but does not specify the Stage-2 geometry and Stage-3 material SLATs. Running the sparse sampler under c_{tgt} can also alter non-edited tokens, as changes in the DINOv3 image-conditioning features propagate through cross-attention. We resolve this by letting the sampler itself identify which tokens are edit-sensitive: a twin forward under c_{src} on the same scaffold and noise reveals, per token, whether the output changes; tokens that agree across the twin are pulled back toward the source encoding, while disagreeing tokens are left untouched.

Why FlowEdit coupling is unavailable. The coupling (3) requires both ODE branches to share the same token space, yet $\mathcal{C}_{\text{tgt}}$ and $\mathcal{C}_{\text{src}}$ generally differ in voxel support after an add or remove edit, making direct coupling impossible. Even when supports overlap, *coordinate coincidence does not imply content preservation*, which is why the geometry-based merge of Nano3D [14] fails.

Twin-forward disagreement map. The key insight is that tokens in non-edited regions should remain relatively stable when the condition changes from the source image to the target image, whereas edit-sensitive tokens should exhibit larger differences. This lets the sampler itself draw the preserve/edit boundary without any external mask.

Let $\mathcal{S}^{\text{stage}}_{\theta}$ denote the Stage-2 or Stage-3 sparse-DiT sampler. On the target scaffold $\mathcal{C}_{\text{tgt}}$ we run twin forwards with identical seeded noise ε, schedule, and CFG weight—only the image embedding differs:

$$z_{\text{tgt}} = \mathcal{S}^{\text{stage}}_{\theta}\big(\mathcal{C}_{\text{tgt}};\, \varepsilon,\, c_{\text{tgt}}\big), \qquad z^{\text{twin}}_{\text{src}} = \mathcal{S}^{\text{stage}}_{\theta}\big(\mathcal{C}_{\text{tgt}};\, \varepsilon,\, c_{\text{src}}\big). \tag{10}$$

Per-token disagreement $d_i = \|z_{\text{tgt}}[i] - z^{\text{twin}}_{\text{src}}[i]\|_2$ is mapped to a preserve-confidence via robust quantile clipping: $p_{\text{flow}}[i] = 1 - \text{clip}\big((d_i - q_\alpha(d))/(q_\beta(d) - q_\alpha(d)),\, 0,\, 1\big)$ with $(\alpha, \beta){=}(0.05, 0.95)$. Tokens producing nearly identical features under both conditions receive $p_{\text{flow}} \approx 1$ and are safe to preserve.

Residual injection. Let $z^{\text{enc}}_{\text{src}}$ be the SC-VAE encoding of the source asset on $\mathcal{C}_{\text{src}}$ and $\mathcal{I} = \mathcal{C}_{\text{tgt}} \cap \mathcal{C}_{\text{src}}$ the integer-voxel intersection. On $\mathcal{I}$ we form a norm-clipped residual $r_i = \text{clip}_\tau(z^{\text{enc}}_{\text{src}}[i] - z_{\text{tgt}}[i])$ and blend:

$$z[i] \;=\; z_{\text{tgt}}[i] \;+\; \lambda\, p_{\text{flow}}[i]\, \mathbf{1}[p_{\text{flow}}[i] \geq \vartheta]\, r_i \quad (i \in \mathcal{I}), \qquad z[i] = z_{\text{tgt}}[i] \quad (i \in \mathcal{C}_{\text{tgt}} \setminus \mathcal{I}), \tag{11}$$

with scale $\lambda \in (0, 1]$ and threshold $\vartheta \in [0, 1)$. Edit-only tokens ($i \in \mathcal{C}_{\text{tgt}} \setminus \mathcal{I}$) stay at the target branch, protecting fresh geometry or material; on intersection tokens the indicator zeroes the update wherever the twin disagrees, while agreeing tokens are softly retracted toward the source encoding with strength governed by p_{flow}.

We apply (10)–(11) once per SLAT (geometry and material). Since the main target forward is bit-identical to the c_{tgt} twin, **TAR adds exactly one extra sparse-DiT forward per stage**. The full pipeline (RASI + PMG on Stage 1, TAR on Stages 2–3) is given in Algorithm 1 (Appendix).

4 Experiments

We evaluate VS3D on an Objaverse-derived benchmark comprising 50 assets and 50 edit cases. Benchmark construction, metrics, and implementation details are provided in Appendix A.1. VS3D uses a fixed set of hyperparameters across all edits in this benchmark.

4.1 Baselines

Baselines. We compare against six methods grouped by whether they require a user-supplied 3D mask. *Mask-based:* **TRELLIS 2.0** [7] (mask-conditioned native repaint), **VoxHammer** [8] (inver-

Table 1: **Quantitative comparison** on our Objaverse-derived editing benchmark. All metrics are averaged over the same set of edits. The *Mask-free* column marks methods that do *not* require a user-supplied 3D mask (✓) versus those that do (×). Best per column is in **bold**.

Method	Mask-free	PSNR↑	SSIM↑	LPIPS↓	DINO-I↑	CLIP-T↑	CD↓
TRELLIS 2.0 [7]	×	17.30	0.793	0.288	0.720	0.278	0.089
VoxHammer [8]	×	18.12	0.786	0.155	0.809	0.283	**0.024**
Nano3D [14]	✓	19.04	0.801	0.242	0.715	0.285	0.096
Instant3dit [11]	×	14.96	0.684	0.229	0.609	0.267	0.053
Edit360 [15]	✓	16.74	0.830	0.299	0.501	0.245	0.173
VecSet-Edit [10]	×	20.40	0.829	0.199	0.763	0.212	0.051
Ours	✓	**22.51**	**0.867**	**0.145**	**0.841**	**0.294**	0.034

sion + KV-replacement), **Instant3dit** [11] (feed-forward multi-view inpainting, 2D-lift), and **VecSet-Edit** [10] (training-free editing on TripoSG). All mask-based methods receive the same human-annotated mask. *Mask-free:* **Nano3D** [14] (FlowEdit + connectivity-based Voxel/SLat-Merge) and **Edit360** [15] (video-diffusion 2D-edit propagation around a 360° trajectory). For fair comparison, we re-implement VoxHammer and Nano3D, both originally built on TRELLIS 1.0, on the same TRELLIS 2.0 backbone.

4.2 Main comparison

Quantitative comparison. Table 1 summarises the results on our benchmark. VS3D achieves the highest PSNR, SSIM, DINO-I, and CLIP-T and the lowest LPIPS among the compared methods. Its CD is second only to that of the mask-based VoxHammer.

To assess generalization beyond our benchmark, we additionally evaluate VS3D on Edit3D-Bench. We rerun VoxHammer, Nano3D, and VS3D under the same evaluation protocol. VS3D ranks first on five of the six metrics; detailed results are reported in Appendix A.2.

In a user study with 21 participants on challenging editing cases, VS3D receives the highest preference share in comparisons with Nano3D and VoxHammer for edit fidelity, identity preservation, and overall quality; details are provided in Appendix A.5.

Qualitative comparison. As shown in Fig. 2, our method produces clean edits while faithfully preserving the geometry and texture of non-edited regions. For elementary operations such as *add* and *remove*, our framework already achieves high-fidelity results with precise identity preservation. More importantly, it handles compound edits that simultaneously add and remove content, where the compared baselines struggle to achieve both the requested changes and source preservation. For *deformation*-style edits that resemble replacement, our method also delivers competitive results, maintaining structural coherence throughout the transformation. Additional large-scale qualitative results are provided in Appendix A.8.

4.3 Ablation studies

We ablate the proposed modules by progressively adding them on top of the base FlowEdit pipeline. Fig. 3 shows two representative cases (rows 1–2 and rows 3–4). For each case, the first column displays the source asset (front and side views) together with the editing pair; subsequent columns incrementally activate each module: FlowEdit alone, +RASI, +RASI+PMG, and +RASI+PMG+TAR (full model).

FlowEdit alone suffers from severe voxel-occupancy drift on non-edited regions. Adding RASI anchors the source trajectory and visibly corrects this drift—the moose's hat shape and position shift back toward the source, and the robot's torso largely recovers its original geometry—yet the edited region now lacks sufficient editing strength. Introducing PMG amplifies the edit signal: the blue hat and the red Christmas hat both emerge clearly, while changes in non-edited areas remain well suppressed. At this stage, however, fine-grained details (e.g. material and texture) of the non-edited region still deviate noticeably from the source, because Stages 2–3 are generated via forward sampling without explicit source-information injection. Finally, TAR injects source-asset details into these later stages: the moose's trousers and body colour more closely match the source, and

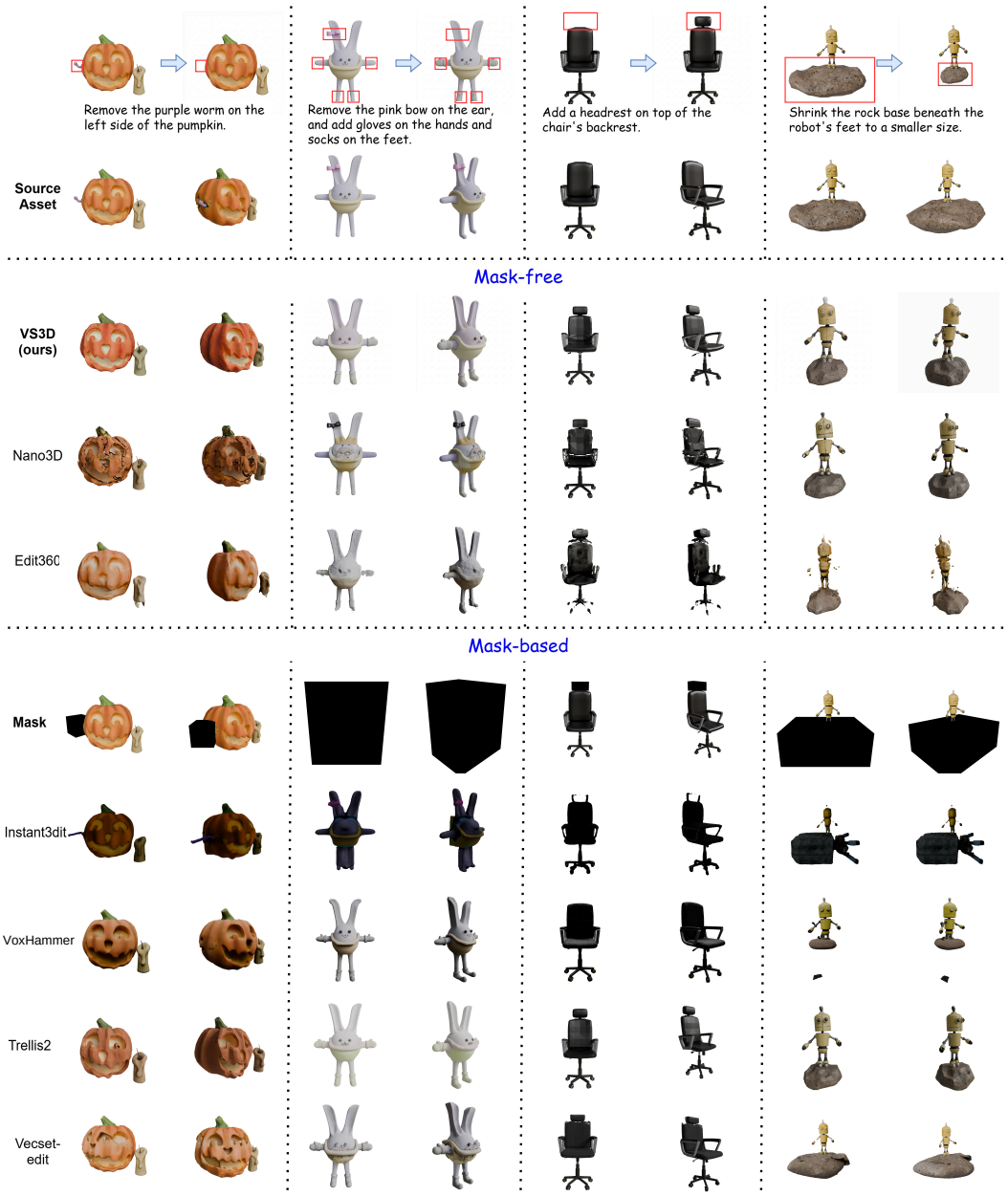


Figure 2: **Qualitative comparison.** Among the mask-free methods compared in Fig. 2, ours is the only approach that jointly preserves the non-edited region and produces high-quality edits in the illustrated cases. Red boxes highlight the edited regions for easier visual inspection and are not part of the model input. In the Mask row, black regions indicate the user-provided editing mask required by mask-based methods.

surface textures on the robot's torso are faithfully restored. Figure 4 visualizes the geometry and material twin-disagreement maps for three edits; additional cases are provided in Appendix A.7.

To quantify the contribution of each module, we progressively add RASI, PMG, and TAR to the FlowEdit pipeline. Table 2 reports the resulting performance: RASI produces the largest gains in preservation-related metrics, while PMG increases CLIP-T from 0.278 to 0.292. Adding TAR further improves PSNR, LPIPS, and CD over the RASI+PMG configuration, consistent with its role in the sparse geometry and material stages. The gains are not uniform across all metrics: RASI alone

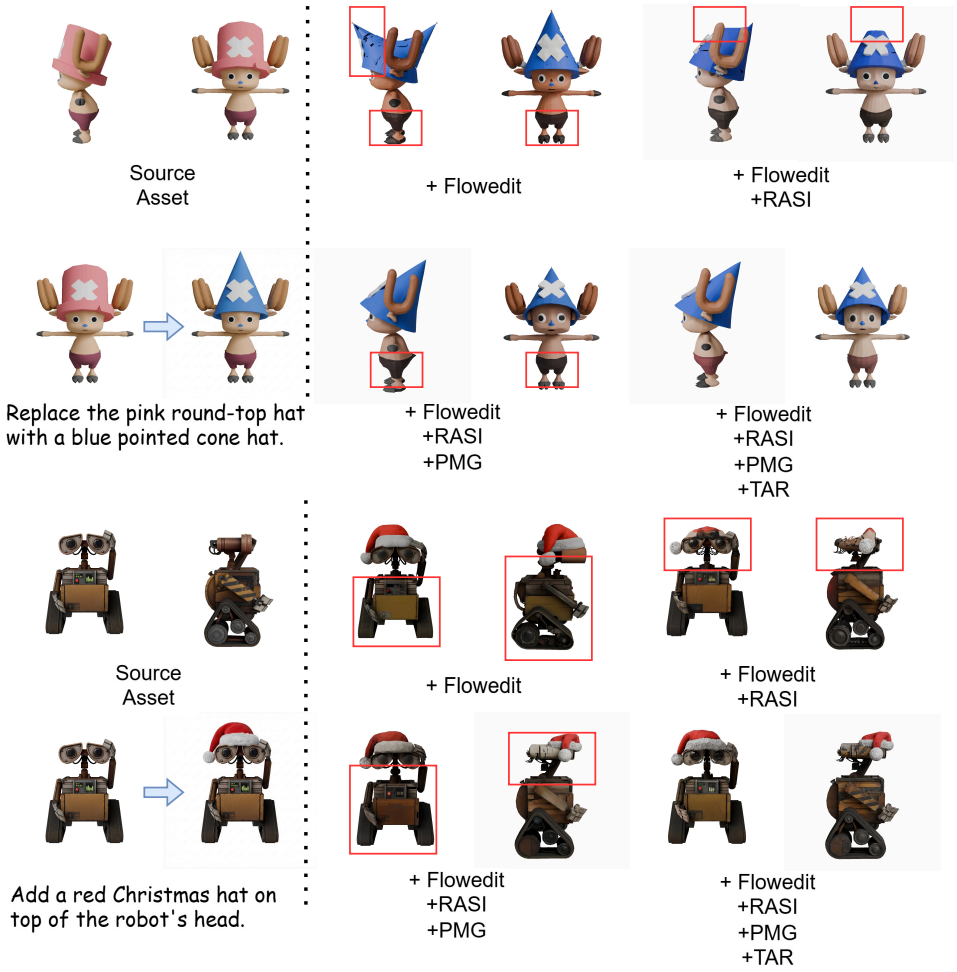


Figure 3: **Ablation study.** Each module is progressively added and resolves a distinct failure mode.

achieves the lowest LPIPS and CD and the highest DINO-I, whereas the full model achieves the highest PSNR and CLIP-T while matching the best SSIM. These results highlight the complementary roles of the modules and the trade-off between edit fidelity and source preservation.

Table 2: **Quantitative ablation of the proposed modules.** Modules are added cumulatively to FlowEdit. Best values in each column are shown in bold.

Configuration	PSNR↑	SSIM↑	LPIPS↓	DINO-I↑	CLIP-T↑	CD↓
FlowEdit (baseline)	20.18	0.804	0.166	0.813	0.225	0.063
+ RASI	22.01	0.856	**0.142**	**0.848**	0.278	**0.032**
+ RASI + PMG	22.43	**0.867**	0.146	0.845	0.292	0.036
Full	**22.51**	**0.867**	0.145	0.841	**0.294**	0.034

5 Discussion

VS3D achieves strong identity preservation in non-edit regions: RASI anchors the source trajectory to suppress occupancy drift, while PMG amplifies the edit signal only where needed. In the geometry and material stages, TAR injects source-encoded residuals into tokens identified as suitable for preservation through twin-branch agreement. However, the inferred preserve/edit partition does not guarantee exact preservation: small changes in geometry or material may remain outside the

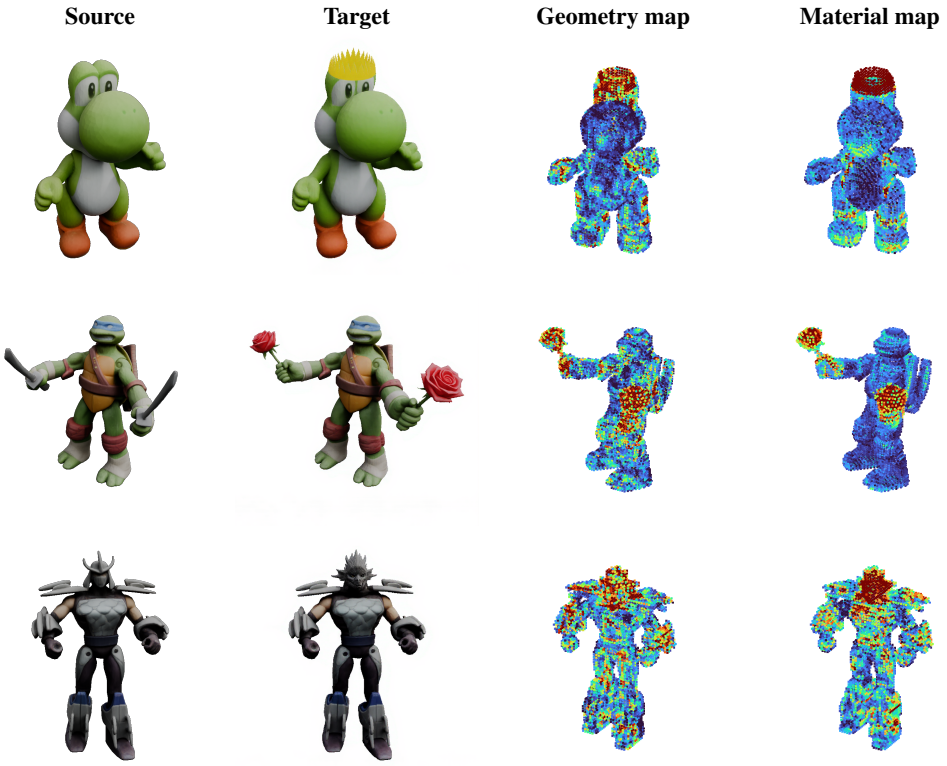


Figure 4: **TAR twin-branch disagreement maps on three Edit3D-Bench cases.** Rows show adding a crown, adding roses, and replacing a head with a dragon helmet. Columns show the source asset, the single-view target condition, and token-wise disagreement maps for geometry and material. Warmer colors indicate larger differences between source- and target-conditioned outputs; cooler colors indicate closer agreement. TAR uses this agreement to weight source-residual injection at overlapping source and target coordinates.

intended edit region. VS3D is therefore intended for source-preserving local editing. Edits that replace nearly the entire asset fall outside this scope, as source anchoring may constrain the substantial changes required.

Furthermore, VS3D operates on frozen TRELLIS 2.0 models, and its editing quality depends on the backbone's ability to represent and generate the requested geometry and materials. Limitations of the backbone can therefore carry over to the edited results. Despite this dependence, VS3D demonstrates that high-quality shape and texture preservation can be achieved through ODE-space modifications without external 3D edit masks, inversion, or backbone fine-tuning. By combining source preservation with targeted editing in a training-free framework, VS3D offers a practical approach to local 3D asset editing. We hope this perspective motivates further research into source-preserving control of pretrained 3D generators.

6 Conclusion

We presented VS3D, a training-free, mask-free framework for high-fidelity local 3D asset editing that operates entirely within the velocity space of a frozen rectified-flow generator. By composing three lightweight ODE-level interventions—RASI for trajectory anchoring, PMG for selective edit amplification, and TAR for geometry and material preservation—VS3D achieves strong identity preservation and faithful editing without masks, inversion, or fine-tuning. Experiments on diverse object categories and edit types, together with evaluations on Edit3D-Bench and a user study, support its effectiveness.

Acknowledgements and Disclosures

This work was partly supported by the National Natural Science Foundation of China (Grant No. 62576191) and the Shenzhen Science and Technology Program (ZDCY20250901103533010).

The authors declare no competing interests.

References

[1] R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6038–6047, 2023.

[2] V. Kulikov, M. Kleiner, I. Huberman-Spiegelglas, and T. Michaeli. FlowEdit: Inversion-free text-based editing using pre-trained flow models. *arXiv preprint arXiv:2412.08629*, 2024.

[3] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[4] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[5] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. E. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. Jégou, P. Labatut, and P. Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

[6] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, and J. Yang. Structured 3D latents for scalable and versatile 3D generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21469–21480, 2025.

[7] J. Xiang, X. Chen, S. Xu, R. Wang, Z. Lv, Y. Deng, H. Zhu, Y. Dong, H. Zhao, N. J. Yuan, and J. Yang. Native and compact structured latents for 3D generation. *arXiv preprint arXiv:2512.14692*, 2025.

[8] L. Li, Z. Huang, H. Feng, G. Zhuang, R. Chen, C. Guo, and L. Sheng. VoxHammer: Training-free precise and coherent 3D editing in native 3D space. *arXiv preprint arXiv:2508.19247*, 2025.

[9] S. Hu, Y. Wei, F. Zha, Y. Guo, and J. Zhang. Easy3E: Feed-forward 3D asset editing via rectified voxel flow. *arXiv preprint arXiv:2602.21499*, 2026.

[10] T.-F. Hsiao, B.-K. Ruan, Y.-L. Liu, and H.-H. Shuai. VecSet-Edit: Unleashing pre-trained LRM for mesh editing from single image. *arXiv preprint arXiv:2602.04349*, 2026.

[11] A. Barda, M. Gadelha, V. G. Kim, N. Aigerman, A. H. Bermano, and T. Groueix. Instant3dit: Multiview inpainting for fast editing of 3D objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16273–16282, 2025.

[12] Y. Chi, X. Li, Z. Huang, and J. M. Rehg. Vinedresser3D: Agentic text-guided 3D editing. *arXiv preprint arXiv:2602.19542*, 2026.

[13] I. Gat, D. Cohen-Bar, G. Levy, E. Richardson, and D. Cohen-Or. ShapeUP: Scalable image-conditioned 3D editing. *arXiv preprint arXiv:2602.05676*, 2026.

[14] J. Ye, S. Xie, R. Zhao, Z. Wang, H. Yan, W. Zu, L. Ma, and J. Zhu. Nano3D: A training-free approach for efficient 3D editing without masks. *arXiv preprint arXiv:2510.15019*, 2025.

[15] J. Huang, X. Hu, S. Shi, Z. Tian, and L. Jiang. Edit360: 2D image edits to 3D assets from any angle. *arXiv preprint arXiv:2506.10507*, 2025.

[16] Y. Chen, J. Wang, L. Liu, R. Chu, X. Zhang, Q. Tian, and Y. Yang. O-DisCo-Edit: Object distortion control for unified realistic video editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3165–3173, 2026.

[17] Z. Lai, Y. Zhao, H. Liu, Z. Zhao, Q. Lin, H. Shi, X. Yang, M. Yang, S. Yang, Y. Feng, S. Zhang, X. Huang, D. Luo, F. Yang, F. Yang, L. Wang, S. Liu, Y. Tang, Y. Cai, Z. He, T. Liu, Y. Liu, J. Jiang, Linus, J. Huang, and C. Guo. Hunyuan3D 2.5: Towards high-fidelity 3D assets generation with ultimate details. *arXiv preprint arXiv:2506.16504*, 2025.

[18] Z. Li, Y. Wang, H. Zheng, Y. Luo, and B. Wen. Sparc3D: Sparse representation and construction for high-resolution 3D shapes modeling. *arXiv preprint arXiv:2505.14521*, 2025.

[19] S. Wu, Y. Lin, F. Zhang, Y. Zeng, Y. Yang, Y. Bao, J. Qian, S. Zhu, X. Cao, P. Torr, and Y. Yao. Direct3D-S2: Gigascale 3D generation made easy with spatial sparse attention. *arXiv preprint arXiv:2505.17412*, 2025.

[20] Z. Lai, Y. Zhao, Z. Zhao, H. Liu, Q. Lin, J. Huang, C. Guo, and X. Yue. LATTICE: Democratize high-fidelity 3D generation at scale. *arXiv preprint arXiv:2512.03052*, 2025.

[21] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3D objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2023.

[22] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[23] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P. Huang, S. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

[24] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021.

A Technical appendices and supplementary material

A.1 Experimental setup

Benchmark. We evaluate on an Objaverse-derived benchmark containing 50 manually selected assets and 50 edit cases. The assets are stored in GLB format and cover characters, weapons, vehicles, creatures, furniture, and everyday objects. We select assets with complete geometry and topology, clear textures and materials, high render quality, and a well-defined local edit region, while maintaining category diversity. For each case, we manually write an edit instruction, render a source view that clearly exposes the target region, and use Nano Banana 2 to edit this view according to the instruction, producing the target condition image. Each case therefore consists of a source 3D asset, a source-view image, an edit instruction, and a single-view target condition image. The edit operations cover add, remove, replace, scale-up, shrink, and compound edits, with an approximate distribution of 20% / 20% / 20% / 10% / 10% / 20%. Since Objaverse is part of the multi-source corpus used by TRELLIS 2.0, overlap with some source assets is possible. However, all edit instructions and target condition images are newly created, so the evaluated asset-instruction-target tuples do not appear in pretraining. In addition, all TRELLIS-based baselines use the same source assets, edit instructions, target condition images, and TRELLIS 2.0 backbone, so any source-asset familiarity is shared across methods rather than favoring VS3D.

Metrics. We report six metrics that jointly cover the three failure modes observed in 3D editing: **(i) PSNR**↑, **(ii) SSIM**↑, and **(iii) LPIPS**↓ [22], measuring pixel-, structural-, and perceptual-level fidelity of the rendered views against renderings of the source asset in the non-edited regions; **(iv) DINO-I**↑, measuring multi-view semantic alignment to the target-view image via a DINOv2 [23] backbone; **(v) CLIP-T**↑ [24], measuring alignment between rendered views and the textual edit instruction; **(vi) CD**↓ (Chamfer Distance), measuring 3D geometric identity preservation against the source mesh on points sampled outside the human-annotated edit region. PSNR/SSIM/LPIPS/DINO-I/CLIP-T are averaged over 16 canonical views at 512^2.

Implementation details. *Stage-1 (RASI + PMG).* We use $T{=}25$ discretised timesteps, an active window governed by $n_{\max}{=}12$ and $n_{\min}{=}0$, and a noise-sample budget of $S{=}5$. PMG uses $(w, L) = (1.2, 2)$. RASI uses $K{=}3$ inner steps with inner-loop learning rate 10^{-5} and early-stop threshold 10^{-5}. The CFG weights are $\omega_{\text{src}}{=}1.5$ and $\omega_{\text{tgt}}{=}9.0$ over the CFG-active interval $t \in [0.6, 1.0]$. *Stage-2 / Stage-3 (TAR).* Both TAR stages run on the 1024-token pipeline with

its native sparse-DiT sampler defaults. The agreement field p_{flow} is computed with robust quantiles $(\alpha, \beta) = (0.05, 0.95)$. TAR in both the geometry and material stages uses λ=0.5, norm-clip τ=10, and agreement threshold ϑ=0.7, with the residual anchored to the target branch (i.e. $r_i = \text{clip}_\tau(z^{\text{enc}}_{\text{src}}[i] - z_{\text{tgt}}[i])$). The src-encoded latents $z^{\text{enc}}_{\text{src}}$ are obtained by running the frozen SC-VAE encoders on the source GLB at 1024^3; both Stage-2 and Stage-3 twins reuse the main-pipeline target forward for z_{tgt}.

A.2 Edit3D-Bench Evaluation

To assess performance beyond our Objaverse-derived benchmark, we evaluate VS3D on Edit3D-Bench and rerun VoxHammer and Nano3D under the same evaluation protocol. The results are independently reproduced rather than copied from the original papers.

Table 3: Evaluation on Edit3D-Bench. All methods are evaluated under the same protocol.

Method	PSNR↑	SSIM↑	LPIPS↓	DINO-I↑	CLIP-T↑	CD↓
VoxHammer	13.8465	0.7195	0.4103	**0.7014**	0.2964	0.1969
Nano3D	15.2320	0.7544	0.4005	0.6959	0.2900	0.1931
VS3D (Ours)	**15.3454**	**0.7621**	**0.3936**	0.6999	**0.2966**	**0.1834**

VS3D ranks first on five of the six metrics, achieving the best PSNR, SSIM, LPIPS, CLIP-T, and CD. Its DINO-I is 0.0015 lower than VoxHammer.

A.3 Sensitivity to the 2D Editor

For each case, we render a single source view exposing the target region and edit it with the 2D editor to obtain the target condition image. This image specifies the intended edit, while all locality control, identity preservation, and geometry/material updates are performed within the 3D ODE. Thus, "native 3D" refers to where the edit is executed rather than how the target condition is authored. To evaluate sensitivity to the choice of 2D editor, we keep the source views, edit instructions, 3D pipeline, and hyper-parameters fixed, and only replace the editor used to produce the target condition.

Table 4: Sensitivity to the choice of 2D editor used to produce the target condition. All other components are fixed.

2D editor	PSNR↑	SSIM↑	LPIPS↓	DINO-I↑	CLIP-T↑	CD↓
Nano Banana 2 (default)	**22.51**	**0.867**	**0.145**	**0.841**	**0.294**	**0.034**
Nano Banana	22.43	0.860	0.152	0.829	0.290	0.036
Qwen-Image-Edit	22.45	0.865	0.149	0.832	0.289	0.039

The results are close across the three editors: relative to the default, the largest deviations are 0.08 dB in PSNR and 0.012 in DINO-I, while SSIM, LPIPS, CLIP-T, and CD vary by at most 0.007, 0.007, 0.005, and 0.005 respectively. This suggests that VS3D is not overly sensitive to the particular 2D editor used to author the condition.

A.4 Runtime comparison

Table 5 reports the wall-clock time for editing a single asset on a single NVIDIA RTX 4090 GPU. VS3D completes the full three-stage pipeline in 57 s without fine-tuning the backbone.

A.5 User Study

We conducted a user study on challenging editing cases. We selected 20 assets, each paired with an edit instruction, including cases that require substantial structural modifications as well as compound edits that combine multiple operations. We recruited 21 participants, including 8 with backgrounds in 3D-related fields and 13 from other professional backgrounds. Participation was voluntary. Before taking part, participants were

Table 5: Per-asset inference time (seconds) on a single NVIDIA RTX 4090 GPU.

Method	Mask-free	Time (s)
VoxHammer	✗	394
Instant3dit	✗	46.2
Nano3D	✓	143
Edit360	✓	495
VS3D (Ours)	✓	**57**

informed that the study involved viewing and comparing 3D-editing results; we anticipated no risks beyond viewing these outputs. Participants received no compensation.

Each participant evaluated all 20 cases. For each case, participants were shown the edit instruction, the source 2D rendering, the edited 2D target image, a 360-degree rendering video of the source 3D asset, and three anonymised result videos produced by VS3D, Nano3D, and VoxHammer. For each participant-case pair, the three results were randomly assigned the labels A, B, and C. Participants selected the best result separately according to three criteria: edit fidelity, identity preservation, and overall quality. Across 21 participants, 20 cases, and three criteria, the study produced 1,260 best-result choices.

Table 6: User-study results. Each cell reports the number of best-result choices and the corresponding percentage out of 420 participant-case evaluations.

Criterion	**VS3D (Ours)**	Nano3D	VoxHammer
Edit fidelity	**223 (53.1%)**	95 (22.6%)	102 (24.3%)
Identity preservation	**218 (51.9%)**	73 (17.4%)	129 (30.7%)
Overall quality	**223 (53.1%)**	82 (19.5%)	115 (27.4%)

VS3D is selected as the best result in a majority of participant-case evaluations under all three criteria. It leads the strongest baseline by 28.8, 21.2, and 25.7 percentage points for edit fidelity, identity preservation, and overall quality, respectively. These results provide complementary human-preference evidence on challenging edits.

Broader impacts. VS3D may reduce the manual effort required for local 3D asset editing in creative workflows. However, it could also be used to modify copyrighted assets without authorization or create misleading visual content. Users should respect asset licenses and obtain permission where required.

A.6 Full algorithm pseudocode

For completeness, Algorithm 1 summarizes the complete VS3D inference pipeline. It follows the three-stage structure described in Sec. 3: RASI calibration and PMG editing are applied to dense occupancy latents in Stage 1, while TAR performs source-residual injection for geometry and material SLATs in Stages 2 and 3. The pseudocode makes explicit the stage-wise inputs, twin forwards, agreement-field computation, and final decoding process.

Algorithm 1 VS3D: three-stage editing pipeline (RASI + PMG + TAR)

Require: source latent x_{src}; source mesh/asset for SC-VAE encoding; conditions $c_{\mathrm{src}}, c_{\mathrm{tgt}}$; active timesteps $\mathcal{T}_{\mathrm{act}}$; CFG weights $\omega_{\mathrm{src}}, \omega_{\mathrm{tgt}}$; noise samples S; PMG (w, L); RASI $(K, \eta_0, \tau_{\mathrm{es}})$; TAR $(\lambda, \tau, \vartheta)$.

▷ **Stage 1: Occupancy editing (RASI + PMG)**
▷ Phase 1: RASI calibration
$z^{\mathrm{edit}} \leftarrow x_{\mathrm{src}}$
for $t \in \mathcal{T}_{\mathrm{act}}$ in descending order **do**
  initialise $\phi_t \leftarrow \phi_0$; set η_t by linear anneal
  for K inner steps **do**
    form $z_t^{\mathrm{src}}, z_t^{\mathrm{tgt}}$ via (3); evaluate $\mathcal{L}_{\mathrm{rec}}(\phi_t)$ of (7)
    $\phi_t \leftarrow \mathrm{Adam}(\phi_t, \nabla_{\phi_t}\mathcal{L}_{\mathrm{rec}}, \eta_t)$; **break** if $\mathcal{L}_{\mathrm{rec}} < \tau_{\mathrm{es}}$
  end for
  advance z^{edit} by one Euler step on the source-reconstruction ODE; cache ϕ_t
end for
▷ Phase 2: PMG editing
$z^{\mathrm{edit}} \leftarrow x_{\mathrm{src}}$
for $t \in \mathcal{T}_{\mathrm{act}}$ in descending order **do**
  for $s = 1, \ldots, S$ **do**
    sample $\varepsilon^{(s)} \sim \mathcal{N}(0, \mathbf{I})$; form $z_t^{\mathrm{src}}, z_t^{\mathrm{tgt}}$ via (3)
    compute $\tilde{v}_{\mathrm{src}}, \tilde{v}_{\mathrm{tgt}}$ with CFG using cached ϕ_t; set $v_\Delta^{(s)} = \tilde{v}_{\mathrm{tgt}} - \tilde{v}_{\mathrm{src}}$
  end for
  $\hat{\mu}_S \leftarrow \frac{1}{S}\sum_{s=1}^{S} v_\Delta^{(s)}$; $\hat{\mu}_L \leftarrow \frac{1}{L}\sum_{s=1}^{L} v_\Delta^{(s)}$
  $u \leftarrow \hat{\mu}_S + w\,(\hat{\mu}_S - \hat{\mu}_L)$ ▷ PMG contrast
  $z^{\mathrm{edit}} \leftarrow z^{\mathrm{edit}} + \Delta t\, u$
end for
decode z^{edit} to edited occupancy $\mathcal{C}_{\mathrm{tgt}}$

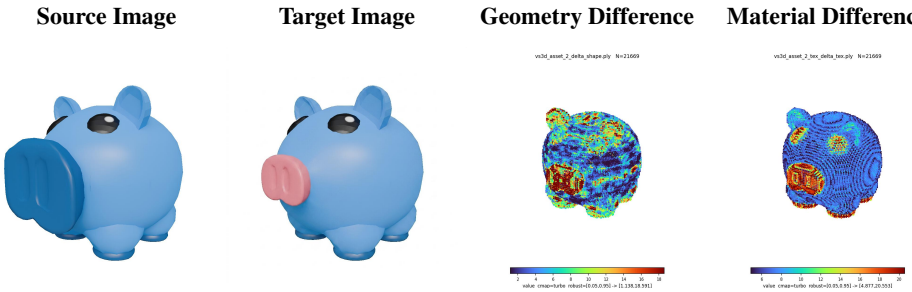
▷ **Stage 2: Geometry SLAT (TAR)**
$z_{\mathrm{tgt}}^{\mathrm{geo}} \leftarrow \mathcal{S}_\theta^{\mathrm{geo}}(\mathcal{C}_{\mathrm{tgt}};\, \varepsilon,\, c_{\mathrm{tgt}})$ ▷ native sparse-DiT forward
$z_{\mathrm{src}}^{\mathrm{geo,twin}} \leftarrow \mathcal{S}_\theta^{\mathrm{geo}}(\mathcal{C}_{\mathrm{tgt}};\, \varepsilon,\, c_{\mathrm{src}})$ ▷ condition-swapped twin
compute per-token agreement $p_{\mathrm{flow}}^{\mathrm{geo}}$ via robust quantile clipping on $\|z_{\mathrm{tgt}}^{\mathrm{geo}}[i] - z_{\mathrm{src}}^{\mathrm{geo,twin}}[i]\|_2$
$z_{\mathrm{src}}^{\mathrm{geo,enc}} \leftarrow \text{SC-VAE}_{\mathrm{geo}}(\text{source asset on } \mathcal{C}_{\mathrm{src}})$
blend via (11) with $(\lambda, \tau, \vartheta)$ on $\mathcal{I} = \mathcal{C}_{\mathrm{tgt}} \cap \mathcal{C}_{\mathrm{src}}$ to obtain z^{geo}

▷ **Stage 3: Material SLAT (TAR)**
$z_{\mathrm{tgt}}^{\mathrm{mat}} \leftarrow \mathcal{S}_\theta^{\mathrm{mat}}(\mathcal{C}_{\mathrm{tgt}}, z^{\mathrm{geo}};\, \varepsilon',\, c_{\mathrm{tgt}})$ ▷ geometry-conditioned
$z_{\mathrm{src}}^{\mathrm{mat,twin}} \leftarrow \mathcal{S}_\theta^{\mathrm{mat}}(\mathcal{C}_{\mathrm{tgt}}, z^{\mathrm{geo}};\, \varepsilon',\, c_{\mathrm{src}})$
compute per-token agreement $p_{\mathrm{flow}}^{\mathrm{mat}}$; encode $z_{\mathrm{src}}^{\mathrm{mat,enc}}$
blend via (11) to obtain z^{mat}

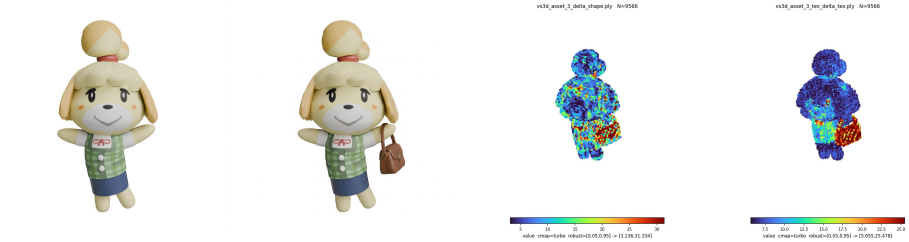
return $(\mathcal{C}_{\mathrm{tgt}},\, z^{\mathrm{geo}},\, z^{\mathrm{mat}})$ ▷ decode to mesh via Flexicubes + texture

A.7 TAR agreement-field visualisation

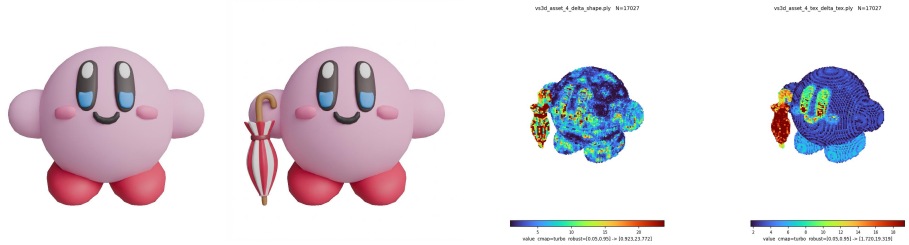
We visualise token-wise differences between the source- and target-conditioned branches in the geometry and material stages. Warmer colours indicate larger differences, whereas cooler colours indicate closer agreement. TAR uses this agreement to weight source-residual injection at overlapping source and target coordinates.



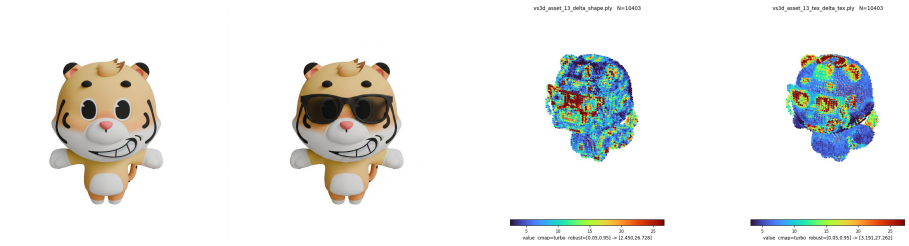
Replace the large dark blue pig snout with a smaller pink snout, keep everything else unchanged.



Add a small handbag to the character's hand, keep everything else unchanged.

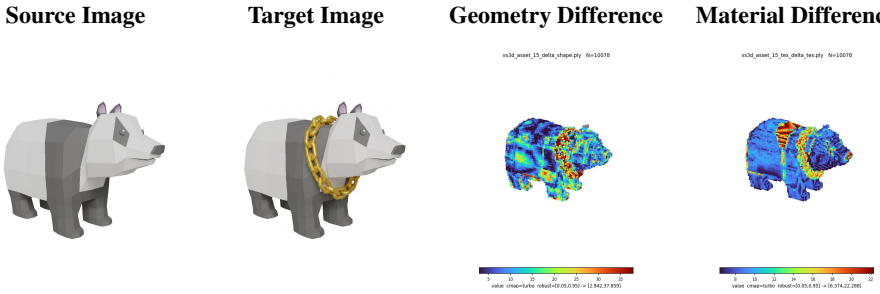


Add a closed red-and-white striped umbrella to the character's hand, keep everything else unchanged.

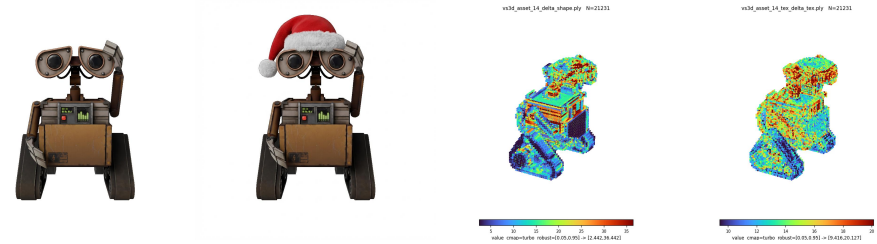


Add a pair of black sunglasses on the character's face, keep everything else unchanged.

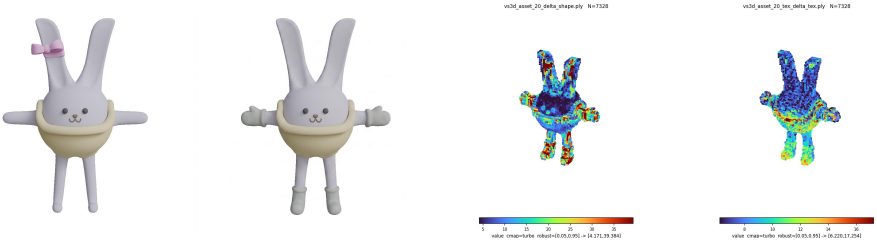
Figure 5: **TAR twin-branch disagreement maps (Part 1).** Columns show the source image, target image, geometry disagreement, and material disagreement. Warmer colours indicate larger differences between the source- and target-conditioned outputs; cooler colours indicate closer agreement. TAR uses this agreement to weight source-residual injection at overlapping source and target coordinates.



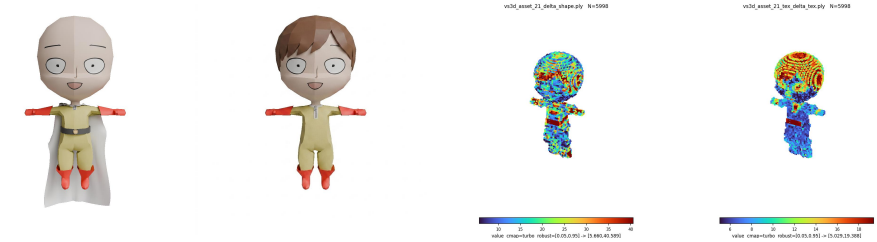
Add a big gold chain necklace around the character's neck, keep everything else unchanged.



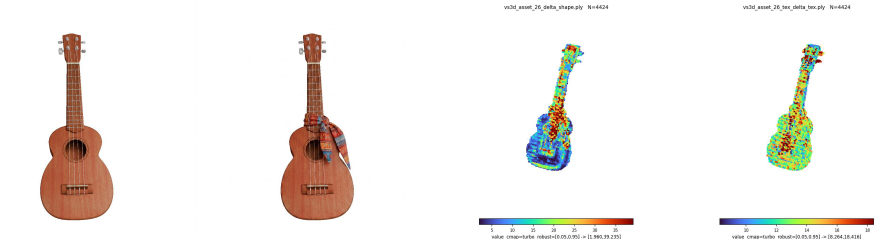
Add a red Christmas hat on top of the robot's head, keep everything else unchanged.



Remove the pink bow on the rabbit's ear, add gloves on its hands and socks on its feet, keep everything else unchanged.



Remove the belt and the white cape from the character, and add side-swept bangs on the character's head, keep everything else unchanged.



Add a scarf tied around the neck of the guitar, keep everything else unchanged.

Figure 6: **TAR twin-branch disagreement maps (Part 2).** Columns show the source image, target image, geometry disagreement, and material disagreement. Warmer colours indicate larger differences between the source- and target-conditioned outputs; cooler colours indicate closer agreement. TAR uses this agreement to weight source-residual injection at overlapping source and target coordinates.

A.8 Large-scale qualitative gallery

To further demonstrate the generality and robustness of VS3D, we present an extended gallery of editing results across a wide variety of 3D assets and edit instructions. These examples span diverse local-edit operations as well as mixed-operation edits, covering diverse object categories including characters, animals, vehicles, and everyday objects. The results confirm that VS3D consistently produces high-fidelity edits while preserving non-edited regions, even under challenging compositional instructions.

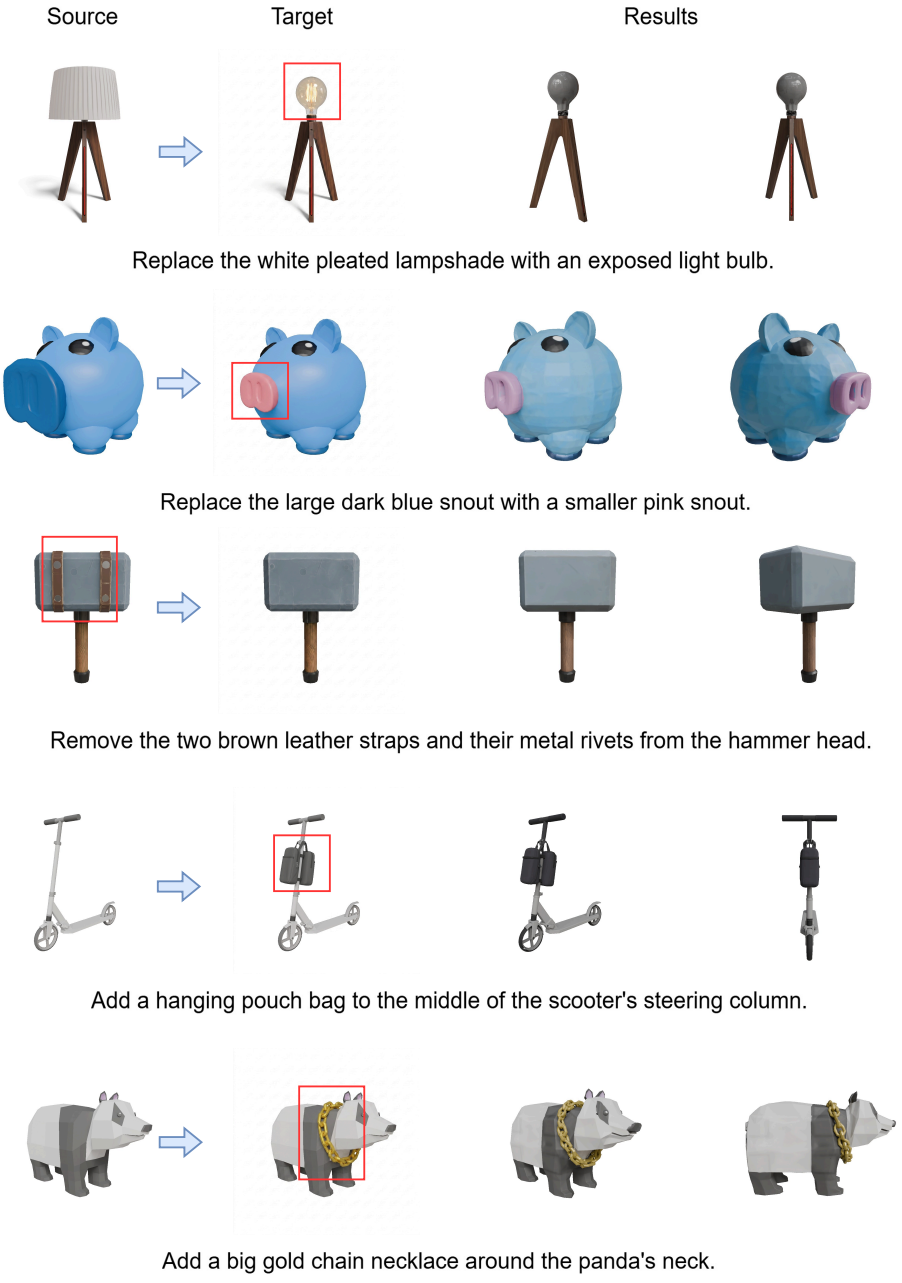


Figure 7: **Extended editing gallery (Part 1).** Large-scale qualitative results of VS3D across diverse assets and edit operations. Each group shows the source asset and the edited result under the corresponding text instruction.

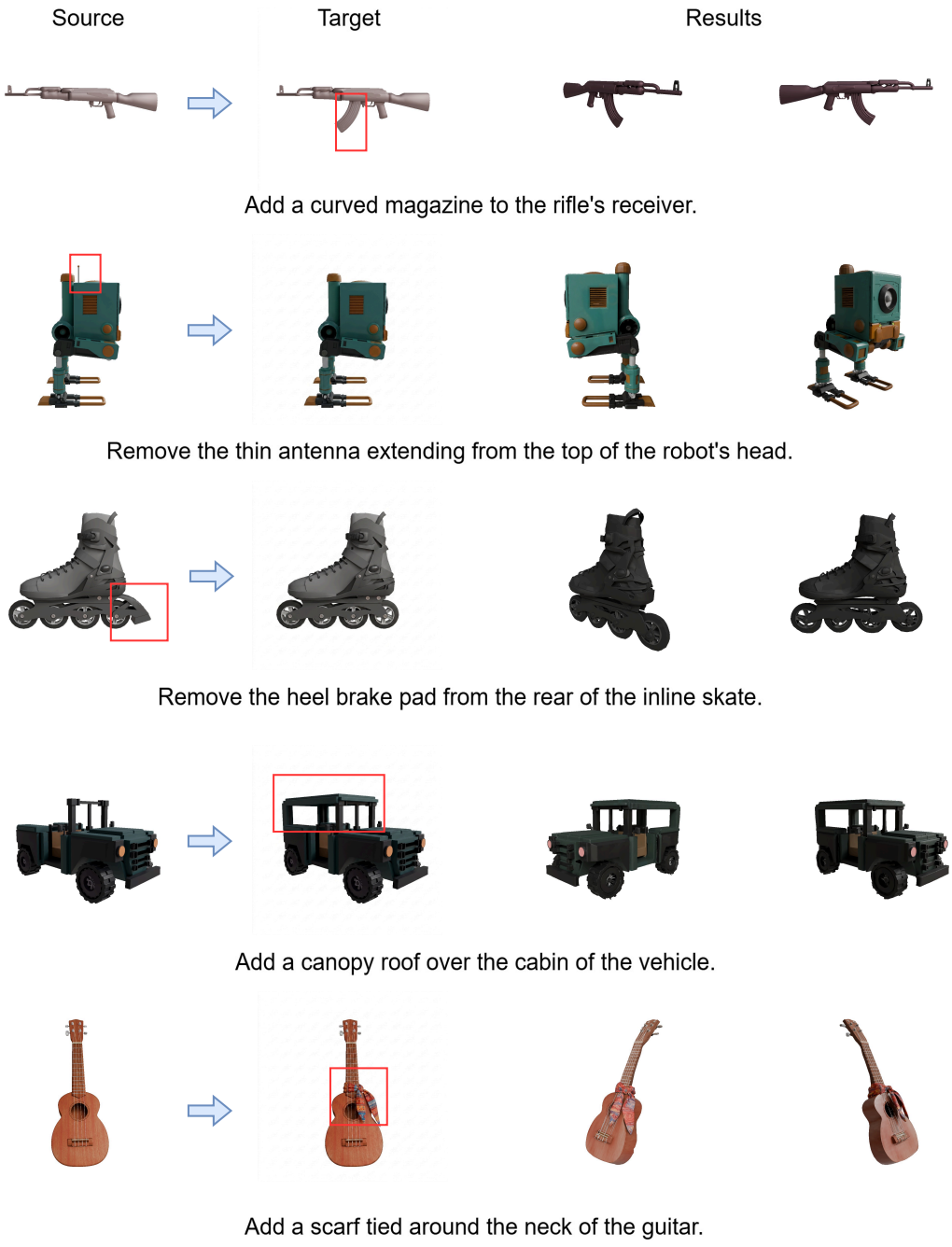


Figure 8: **Extended editing gallery (Part 2).** Additional large-scale results demonstrating VS3D's robustness on mixed-operation edits and complex compositional instructions.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state that VS3D is a training-free, mask-free 3D editing framework operating in velocity space, which is evaluated through the quantitative, qualitative, ablation, benchmark, and user-study experiments in Section 4 and Appendix A.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 discusses residual geometry and material changes in non-edited regions, the source-preserving editing scope, and dependence on the generative capacity of the TRELLIS 2.0 backbone.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper does not include formal theoretical results or proofs; it is a methods paper validated empirically.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Algorithm 1 describes the inference pipeline, and Appendix A.1 reports the benchmark construction, metrics, and implementation settings. Our method builds on the publicly available TRELLIS 2.0 model; implementation and setup instructions are available at `https://github.com/LHaoooo/VS3D`.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The implementation and setup instructions are publicly available at `https://github.com/LHaoooo/VS3D`. The benchmark data are not yet fully public and will be released progressively. Therefore, complete public access to the data needed to reproduce the reported benchmark results is not yet available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 describes the comparisons and ablations, and Appendix A.1 reports the benchmark construction, metrics, and implementation settings. The user-study protocol is provided in Appendix A.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: We report aggregate benchmark metrics and the user-study vote counts and percentages. We do not report confidence intervals or statistical significance tests.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Appendix A.4 reports per-asset runtime on a single NVIDIA RTX 4090 GPU for VS3D and four baselines. Peak memory usage and the total compute used across all experiments are not reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research includes a voluntary preference study with 21 participants. Participants were informed about the study procedure before participating, and no compensation was provided.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader-impact paragraph in Appendix A.5 discusses benefits for creative workflows, potential unauthorized asset modification and misleading content, and responsible use respecting asset licenses and permissions.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: VS3D does not release new pre-trained models or scraped datasets; it operates on an existing publicly available model (TRELLIS 2.0).

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: TRELLIS 2.0 (MIT License) and Objaverse (ODC-By License) are properly cited. All 3D assets used are from publicly available datasets with permissive licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: The paper introduces an Objaverse-derived editing benchmark and releases the VS3D implementation, but no new pretrained model weights. Appendix A.1 documents the benchmark construction. The benchmark data will be released progressively; a complete public benchmark package with accompanying documentation is not yet available.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [No]

Justification: Appendix A.5 describes the study protocol and states that participants received no compensation. The original participant instructions and interface screenshots were not retained, so they cannot be provided.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

    Justification: The study involved voluntary preference judgments on rendered 3D-editing results. Participants were informed of the study procedure beforehand and received no compensation. We anticipated no risks beyond viewing these outputs. The supervising faculty member agreed to the study being conducted; however, no formal IRB or equivalent institutional ethics review was obtained.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as a component in the core methodology of VS3D.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.