PACE3D: CONTROLLING STRUCTURAL PROGRESS IN 3D EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training-free 3D editors typically stop at a fixed sampling endpoint, implicitly assuming elapsed time implies structural completion. This can fail in hierarchical 3D generators: sparse-structure latents evolve continuously, while decoded support changes discretely as occupancy logits cross thresholds. Sampling can therefore end with incomplete target support, unrecoverable by downstream feature generation. We call this failure *structural progress–clock mismatch*. **Pace3D** uses the structure decoder as an online progress sensor. A target-conditioned proxy provides a structural reference; when decoded structure lags behind its reference phase, Pace3D rewinds the clock while retaining the current latent. Source Alignment stabilizes the extended trajectory, while Cross-Condition Consensus preserves unchanged source features. Diagnostics show that larger terminal phase lag corresponds to more missing target structure. On Eval3DEdit, Pace3D further achieves the best target alignment while maintaining strong source fidelity.

1 INTRODUCTION

Editing an existing 3D asset requires realizing the requested geometric and appearance changes while preserving its identity and all unrelated content. Training-free methods are especially attractive, as they reuse pretrained 3D priors without task-specific training or per-asset optimization (Li et al., 2026; Ye et al., 2025; Zhou et al., 2026). Many scalable 3D priors (Xiang et al., 2025; 2026; Lai et al., 2025), however, are hierarchical: they first generate a sparse structural (SS) latent that determines where geometry exists, and then synthesize structured-latent (SLat) features only on active voxels. This makes the SS stage a one-way bottleneck. Once its support is fixed, SLat can refine existing voxels but cannot recover a requested component for which no voxel was activated.

Existing training-free flow editors largely focus on how the source-to-target trajectory is constructed or constrained, while retaining a predetermined sampling schedule (Kulikov et al., 2025; Ye et al., 2025). Their stopping rule is therefore open-loop in that sampling ends at $t = 0$ without checking whether the requested geometry has formed. This can fail because the SS latent evolves continuously, whereas decoded support changes only when occupancy logits cross discrete thresholds. Since the learned field only approximates ideal source-to-target transport (Li et al., 2025b; Kim et al., 2025), some logits can remain just below threshold when the schedule ends. As Fig. 1 shows, the clock can finish while an entire target component remains incomplete. The sampler tracks elapsed integration, not realized geometry. We call this failure the *structural progress–clock mismatch*.

This mismatch is more consequential than an ordinary residual error, since thresholding can turn a small continuous gap into a semantically large support failure, e.g., an incomplete headphone. Once incomplete support enters SLat, the missing structure is irreversible. So reaching $t = 0$ certifies elapsed integration, not structural completion. The key question is not only *how* an editing trajectory should move, but whether it has moved *far enough*. Finer discretization provides no completion signal, while uniformly extending all trajectories wastes computation and increases off-target drift. A reliable editor should instead observe realized structure and continue only when necessary.

Correcting the progress–clock mismatch is necessary, but simply extending the trajectory is not enough for reliable editing. A practical solution must address three linked requirements. First, it must determine **when to continue**: without access to the true target support or future target-directed progress, the editor must still recognize when realized structure lags behind the sampling clock and the remaining horizon is likely insufficient. Second, once additional horizon is allocated, it must

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

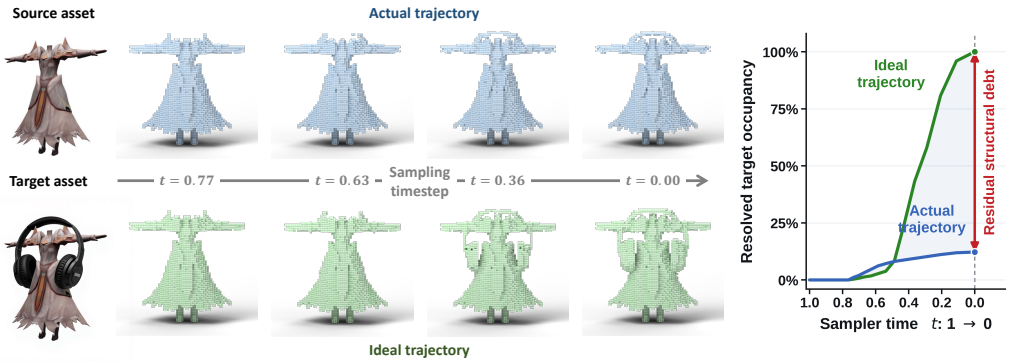


Figure 1: **Sampling clock can finish before the geometry does.** The fixed-clock trajectory (top) reaches $t = 0$ while the requested headphones remain structurally incomplete, illustrating the *structural progress–clock mismatch*. An ideal trajectory (bottom), constructed from the paired target asset, completes the target support within the same horizon. As SLat generation operates on the finalized SS support, structure missed during SS editing cannot be recovered downstream.

determine **how to continue safely**: the extra updates should resolve unfinished target events without accumulating off-target geometry or turning under-editing into over-editing. Third, structural completion alone does not guarantee a faithful asset. When the corrected support enters the SLat stage, the editor must determine **what to preserve**, retaining source features at genuinely unchanged locations without copying obsolete content into edited or newly created geometry. Together, these requirements form a causal chain: detect and compensate structural under-progress, stabilize the resulting corrective updates, and convert the completed support into a source-faithful edit.

Contribution. We introduce **Pace3D**, a training-free framework with three components. A *Structural-Progress Controller* decides when the SS trajectory needs additional horizon; *Source Alignment* stabilizes the resulting updates; and *Cross-Condition Consensus* identifies which source SLat features can be safely restored. They together solve the structural progress–clock mismatch.

When to continue: Structural-Progress Controller. We define *structural debt* as the decoder-logit change needed to complete an occupancy event; completion requires sufficient target-directed progress over the remaining horizon to clear it. Because the target support and future progress are unknown, Pace3D uses a target-conditioned proxy as a *structural ruler*. A source-to-proxy path provides reference phases for locating the current structure. When the matched phase lags behind the sampling clock, Pace3D rewinds the clock while retaining the current latent, preserving realized progress while granting unfinished events more horizon. This state-dependent controller turns the structure decoder into an online progress sensor before missing support becomes irreversible.

How to continue safely: Source Alignment. Additional horizon can complete target structure, but can also accumulate geometric leakage. Pace3D therefore applies a weak contraction toward the source latent along the SS trajectory. Motion consistently reintroduced by the target field can still accumulate, whereas transient deviations are attenuated. Rather than estimating an edit mask or freezing predefined regions, Source Alignment regulates motion over the controller's variable horizon, preventing additional completion opportunity from becoming uncontrolled source drift.

What to preserve: Cross-Condition Consensus. A completed support does not determine which SLat features should come from the source. Pace3D compares source- and target-conditioned predictions at the same noisy state and aggregates disagreement across timesteps. Strong disagreement keeps features target-driven, while agreement supports source restoration. Fusion is limited to shared coordinates and weighted by this evidence; new voxels remain target-generated. So local preservation is inferred from model-internal temporal evidence without user masks.

Experiments support our diagnosis and the resulting design. Structural phase lag predicts terminal under-completion, motivating decoded geometry as a feedback signal (Fig. 3). Under matched compute, Pace3D achieves the best target alignment while maintaining competitive source fidelity (Table 2). Ablations in Table 3 confirm the complementary roles of the controller, Source Alignment, and Cross-Condition Consensus. Together, these results support progress-aware horizon control as a principled and effective strategy for training-free 3D editing.

2 Related Work

3D Generative Priors. Early text-to-3D methods rely on per-instance optimization with 2D diffusion priors (Poole et al., 2023; Lin et al., 2023; Wang et al., 2023), while recent methods learn scalable native 3D representations and priors (Zhang et al., 2023; Zhao et al., 2023; Wu et al., 2024; Zhang et al., 2024; Zhao et al., 2025; Xiang et al., 2025; 2026). Hierarchical generators (Xiang et al., 2025; 2026) predict sparse support before synthesizing local geometry and appearance, making support an irreversible interface: downstream stages cannot recover missing geometry. Pace3D turns this interface into inference-time feedback, using the structure decoder as a progress sensor.

3D Editing and Source Preservation. Existing methods optimize 3D representations with 2D guidance (Sella et al., 2023; Wang et al., 2024; Dong et al., 2024; Zhuang et al., 2024), reconstruct from edited views (Qi et al., 2024; Barda et al., 2025; Erkoç et al., 2025; Bar-On et al., 2025; Li et al., 2025a), or edit native 3D latents (Xiang et al., 2025; Li et al., 2026; Ye et al., 2025; Xia et al., 2025; Weng et al., 2026; Zhou et al., 2026). Recent training-free approaches improve source-to-target transport and preservation through inversion-free flow (Kulikov et al., 2025; Ye et al., 2025), feature/voxel merging (Li et al., 2026), or latent anchoring (Zhou et al., 2026). They mainly address *how* an edit evolves and *what* source content is preserved under a fixed schedule. Pace3D instead asks *when* decoded structure is complete and adapts the remaining horizon when it is not.

Adaptive Sampling and Feedback-Guided Editing. Prior work adapts sampling through restart or timestep allocation to improve numerical accuracy and sample quality (Xu et al., 2023; Chen et al., 2024; Sabour et al., 2024), while feedback has been used for semantic stopping or refinement, e.g., CLIP-guided patience stopping in 3D editing and VLM-based verification in image editing (Wu et al., 2026; Bai et al., 2026). Pace3D instead uses the occupancy decoder in 3D editing to detect per-instance structural lag and extend the horizon before missing support becomes irreversible. Its objective is structural completion rather than schedule optimization or semantic early stopping.

3 Structural Progress–Clock Mismatch

Sec. 1 identified a termination failure in which sampling may end before the requested geometry is complete. Here we formalize this mismatch. We first show how continuous latent motion can leave discrete support events unresolved, then characterize each event by the structural debt it still owes and the target-directed progress available to clear it. Finally, we show why the remaining horizon can become insufficient even as time advances toward $t = 0$.

3.1 Continuous Editing, Discrete Support

Let $\boldsymbol{x}^{\mathrm{src}}$ and $\boldsymbol{z}^{\mathrm{src}}$ denote source SS and SLat latents, and let $\boldsymbol{c}_{\mathrm{src}}$ and $\boldsymbol{c}_{\mathrm{tgt}}$ denote source and target conditions. Time runs from $t = 1$ to $t = 0$. Following training-free flow editing (Kulikov et al., 2025), shared-noise source and target model queries define an SS editing field $\boldsymbol{f}_t$, yielding

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boldsymbol{f}_t(\boldsymbol{x}_t). \tag{1}$$

The queries are standard and deferred to Appendix A.1; here, the point is that Eq. (1) moves $\boldsymbol{x}_t$ continuously. The structure exposed to the next stage is instead discrete. Let $D_j(\boldsymbol{x})$ be the occupancy logit at voxel j and κ the activation threshold. The decoded occupancy and support are

$$O_j(\boldsymbol{x}) = \mathbf{1}[D_j(\boldsymbol{x}) \geq \kappa], \qquad \Omega(\boldsymbol{x}) = \{j : O_j(\boldsymbol{x}) = 1\}. \tag{2}$$

Although the latent changes at every solver step, realized geometry changes only when a decoder logit crosses κ. A trajectory may therefore make continuous latent progress without completing the required support change.

This gap becomes irreversible at the stage boundary. Once SS sampling returns $\boldsymbol{x}_0$, SLat generation is restricted to the support $\Omega_0 = \Omega(\boldsymbol{x}_0)$ and can refine active-voxel features but cannot create a requested component whose voxels never became active. SS termination therefore marks the boundary beyond which missing support becomes irreversible, so structural completion must be assessed in decoder space rather than from numerical time alone. Binary occupancy tells us whether an event has occurred, but not how far an unfinished event remains from completion. We next quantify this remaining gap.

3.2 Structural Debt and Target-Directed Progress

Reasoning about completion requires two quantities: how much target-directed decoder change an unfinished event still requires, and how much such useful progress the remaining trajectory can still provide. We first quantify the former. Consider a voxel j whose desired occupancy is $O_j^\star \in \{0,1\}$. Let $s_j = 2O_j^\star - 1$, so $s_j = +1$ denotes addition and $s_j = -1$ deletion. We define its signed target margin and remaining *structural debt* as

$$g_j(\boldsymbol{x}) = s_j\big(D_j(\boldsymbol{x}) - \kappa\big), \qquad m_j(\boldsymbol{x}) = \big[-g_j(\boldsymbol{x})\big]_+. \tag{3}$$

For either addition or deletion, m_j is the remaining logit change needed to cross the threshold in the desired direction, vanishing once the desired side is reached. Structural debt is therefore determined by the current decoded state, not by the sampler clock.

We next consider the second quantity: how much of this debt the remaining trajectory can still clear. Only motion that improves the corresponding target margin counts as useful structural progress. Under decreasing-time sampling, we define the instantaneous target-directed progress as

$$q_j(t) = -\nabla_{\boldsymbol{x}} g_j(\boldsymbol{x}_t)^\top \boldsymbol{f}_t(\boldsymbol{x}_t). \tag{4}$$

The sign is chosen so that $q_j(t) > 0$ means the reverse-time update moves event j toward its target occupancy. For an unfinished event, $g_j(\boldsymbol{x}_t) < 0$ and $m_j(\boldsymbol{x}_t) = -g_j(\boldsymbol{x}_t)$. Applying the chain rule from the current time t to the endpoint gives the exact completion condition

$$\text{event } j \text{ completes by } t = 0 \quad \Longleftrightarrow \quad \int_0^t q_j(\tau)\,\mathrm{d}\tau \geq m_j(\boldsymbol{x}_t). \tag{5}$$

Eq. (5) gives a simple resource-balance view of structural completion, in which the remaining *useful* progress must be sufficient to clear the current structural debt. Here, $m_j(\boldsymbol{x}_t)$ measures how much target-directed decoder change is still needed for event j to cross its occupancy threshold, while $q_j(t)$ measures the instantaneous rate at which the trajectory supplies such change. Crucially, only motion that improves the target decoder margin contributes to completion; latent movement in other directions does not reduce the debt. A trajectory may therefore move substantially, or even exhaust its sampling grid, while the desired event remains unresolved. Thus, *latent movement is not equivalent to structural progress, just as elapsed time is not equivalent to structural completion.*

This condition characterizes the failure but is not directly observable, since the desired occupancy $O_j^\star$ is unknown and future target-directed progress is unavailable in advance. The analysis is therefore diagnostic, revealing what fixed-clock termination fails to observe. Before introducing an observable surrogate, we show why unresolved debt becomes especially problematic late in the trajectory, when the remaining horizon may itself be insufficient.

3.3 When the Remaining Horizon Is Insufficient

A simple upper bound makes the failure mechanism explicit. Suppose that, over the remaining interval $[0,t]$, decoder sensitivity and field magnitude are bounded by $\|\nabla_{\boldsymbol{x}} g_j(\boldsymbol{x}_\tau)\|_2 \leq \bar{L}_j$ and $\|\boldsymbol{f}_\tau(\boldsymbol{x}_\tau)\|_2 \leq \bar{B}$. Then even the most optimistic target-directed progress is bounded by

$$\int_0^t q_j(\tau)\,\mathrm{d}\tau \leq \int_0^t [q_j(\tau)]_+\,\mathrm{d}\tau \leq \bar{L}_j \bar{B}\, t, \qquad m_j(\boldsymbol{x}_t) > \bar{L}_j \bar{B}\, t \implies g_j(\boldsymbol{x}_0) < 0. \tag{6}$$

The implication is deliberately one-sided: if this optimistic upper bound is smaller than current debt, the event cannot reach its target threshold before time runs out. The converse need not hold, as sufficient nominal capacity does not ensure that the learned field uses it in the useful decoder direction. Importantly, $\bar{L}_j \bar{B}\, t \to 0$ as $t \to 0$, while structural debt is state-dependent and need not vanish with the clock. So a late unresolved event can enter a regime where too little horizon remains to complete it. This is the structural progress–clock mismatch in its simplest form. The clock is ensured to finish, but structure is not. Appendix A.1 gives the full time-varying bound and derivation.

This analysis reveals a key observability gap: as a reliable controller must know when decoded structure is falling behind, while neither the true structural debt nor future useful progress is available during editing. So Pace3D replaces the oracle debt–capacity comparison with an observable, proxy-calibrated structural phase, using its lag from the sampling clock to decide when additional horizon is needed. As this intervention adds extra updates, the extended trajectory must also be stabilized before the final support is passed to source-preserving feature generation. We describe them below.

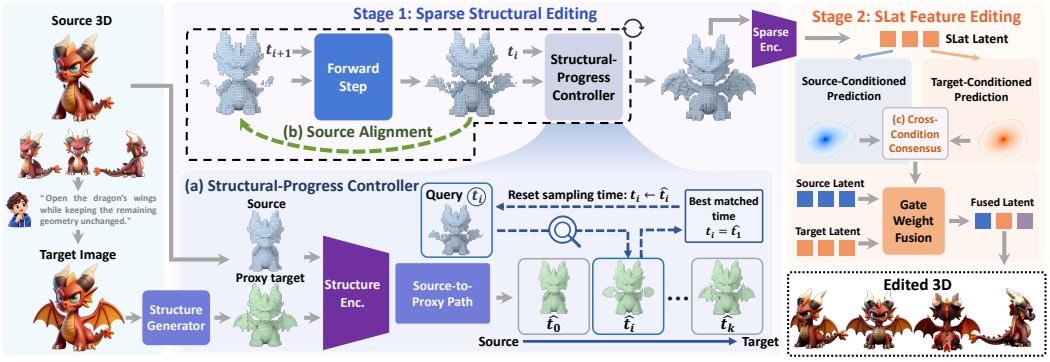


Figure 2: **Pace3D closes the loop between sampling time and decoded structure.** In Stage I, the Structural-Progress Controller locates the current occupancy on a source-to-proxy reference path and rewinds the clock when realized structure lags behind nominal time. Source Alignment stabilizes the resulting extended trajectory. In Stage II, Cross-Condition Consensus restores source features in proportion to how closely source- and target-conditioned responses agree.

4 Pace3D: Decoder-Feedback Horizon Control

Sec. 3 shows that fixed-clock editing lacks an observable completion signal. Motivated by this analysis, Pace3D introduces decoder feedback for adaptive horizon control. Figure 2 summarizes the resulting two-stage procedure. Before editing, Pace3D builds a source-to-proxy structural ruler from the pretrained 3D prior. During SS editing, the Structural-Progress Controller uses decoded occupancy to detect structural lag and extends the horizon only when needed, while Source Alignment stabilizes the extended trajectory. Once the support is fixed, Cross-Condition Consensus determines which source SLat features can be safely restored. The pipeline thus follows a causal sequence: *observe under-progress, correct it safely, and preserve source content on the corrected support*.

4.1 When to Continue: Proxy-Calibrated Structural-Progress Control

A target proxy as a structural ruler. Sec. 3 tells us what a perfect controller would need to know, but the true target support and future useful progress are unavailable. We therefore replace this oracle debt–progress comparison with an observable surrogate for structural progress. Pace3D obtains an approximate target-conditioned SS latent $\boldsymbol{x}^{\text{proxy}}$ from the same pretrained 3D prior (Appendix A.6). The proxy is deliberately not treated as the desired edited asset. Independent generation may alter source identity or unrelated geometry. Its role is much weaker—it only provides an ordering of the structural changes suggested by the target image. We connect the source and proxy with

$$P_t = \frac{t}{t_{\text{start}}}\boldsymbol{x}^{\text{src}} + \left(1 - \frac{t}{t_{\text{start}}}\right)\boldsymbol{x}^{\text{proxy}}, \qquad t \in [0, t_{\text{start}}], \tag{7}$$

where t_{start} denotes the start of SS editing, such that $P_{t_{\text{start}}}$ is the source and P_0 the proxy. To see why this path can serve as a progress ruler, consider a locally monotone reference logit $\ell_j(t) = D_j(P_t)$ that crosses κ at τ_j, and define its proxy-side debt as in Eq. (3). A first-order expansion around the crossing gives

$$\widehat{m}_j^{\text{ref}}(P_t) \approx \left|\dot{\ell}_j(\tau_j)\right|[t - \tau_j]_+. \tag{8}$$

The derivation is given in Appendix A.2. Thus, near a structural event, unresolved reference debt is encoded by a *phase gap*, with an event further behind its crossing corresponding to a larger remaining time on the reference path. This does not imply that the edited trajectory must follow the proxy path. Rather, it motivates a more robust question than estimating noisy online debt-reduction rates: *which decoded structural phase on the reference ruler most resembles the current edit?*

Locating the edit in decoder space. The reference path provides a progress scale; the next step is to locate the current edit on that scale. A direct projection in the full latent space is fragile because the structural edit may occupy only a small subspace, while unrelated latent variation and identity differences from proxy generation can dominate the distance. Since the failure identified in Section 3

occurs when occupancy events remain unresolved, Pace3D measures progress in that same decoder space. It decodes the reference path once on the sampler grid and caches

$$\mathcal{C} = \left\{ \left(t_k, O(P_{t_k})\right) \right\}_{k=0}^{K}, \qquad \widehat{k} = \arg\min_k \left\| O(\boldsymbol{x}_t) \oplus O(P_{t_k}) \right\|_1, \qquad \widehat{t} = t_{\widehat{k}}. \tag{9}$$

The Hamming lookup finds the reference support closest to the current decoded geometry, ignoring latent changes that do not affect occupancy. Only voxels that change along the reference path are informative for phase matching. Unchanged voxels contribute the same constant to every candidate distance and therefore do not affect the matched phase. Appendix A.2 provides the formal argument.

Define the structural lag as $\delta(t) = \widehat{t} - t$. Since sampling proceeds from larger to smaller time, $\delta(t) > 0$ means that the decoded geometry resembles an *earlier* reference phase than the sampling clock suggests, with the clock indicating more progress than is visible in the structure. Thus, $\delta(t)$ provides an observable signal of structural under-progress. Because the proxy is approximate and the editing field need not follow the reference at the same rate, we intervene only when $\delta(t) > \theta$, using θ as a robustness margin against small phase mismatches.

Rewinding the clock. When $\delta(t) > \theta$, Pace3D retains the current latent and resets only the sampling time,

$$(\boldsymbol{x}_t, t) \longmapsto (\widetilde{\boldsymbol{x}}_{\widehat{t}}, \widehat{t}) = (\boldsymbol{x}_t, \widehat{t}). \tag{10}$$

All source and target flow queries are rebuilt at $\widehat{t}$ while the current latent is retained. Thus, the structural state and debt remain unchanged, but the interval for future target-directed progress expands from $[0, t]$ to $[0, \widehat{t}]$. Pace3D therefore *rewinds the clock, not the state*: realized structure is preserved, while unfinished occupancy events gain more opportunity to cross their thresholds. Importantly, $\widehat{t}$ serves as a *horizon-allocation control variable* for the editing process. Rewinding neither removes the current debt nor creates a useful direction; it only gives the existing editing field more time to reduce it. Appendix A.3 formalizes this distinction and details query reconstruction, and repeated-reset safeguards.

The controller turns the structure decoder from a final occupancy renderer into an online progress sensor, making the sampling horizon instance- and state-dependent. This determines *when to continue*; the next question is how to prevent the added opportunity from amplifying off-target motion.

4.2 How to Continue Safely: Source-Aligned Horizon Extension

A longer horizon gives missing occupancy events more opportunity to cross their thresholds, but it also gives unintended motion more opportunity to accumulate. Pace3D therefore augments each SS update with a weak contraction toward the source latent. For consecutive times $t_{n+1} < t_n$,

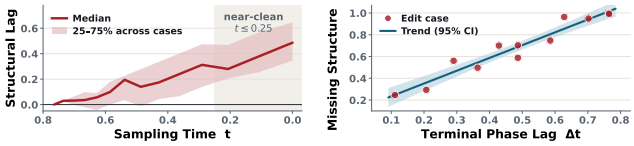
$$\boldsymbol{x}_{t_{n+1}} = \boldsymbol{x}_{t_n} + (t_{n+1} - t_n)\boldsymbol{f}_{t_n}(\boldsymbol{x}_{t_n}) + \eta\big(\boldsymbol{x}^{\mathrm{src}} - \boldsymbol{x}_{t_n}\big), \qquad 0 < \eta \ll 1. \tag{11}$$

The flow term supplies target-directed editing motion, while the contraction makes accumulated displacement behave like a leaky memory, repeatedly discounting transient deviations while allowing directions consistently reinforced by the target field to accumulate. Source Alignment therefore need not infer an edit mask or explicitly separate intended from unintended motion. Instead, it raises the temporal persistence required for a deviation to survive. Under bounded off-target increments, Appendix A.4 shows that their accumulated effect remains bounded over an extended horizon rather than growing linearly. Keeping η small prevents source attraction from dominating the target field while reducing the chance that transient motion becomes a permanent occupancy change.

Thus, the controller provides additional opportunity for structural completion, while Source Alignment prevents that opportunity from becoming uncontrolled drift. Stage I has now answered the first two questions—*when* to continue and *how* to continue safely. The resulting support, however, still does not determine which local features should be inherited from the source. This final preservation decision is handled in the SLat stage.

4.3 What to Preserve: Cross-Condition Consensus

Let $\Omega_0 = \Omega(\boldsymbol{x}_0)$ be the final support passed to SLat generation, and $\mathcal{I} \subseteq \Omega_0$ the coordinates shared with the source. Newly activated coordinates have no source counterpart and remain target-generated. At shared coordinates, restoring the source may suppress a desired edit, while following



(a) Structural lag over time. (b) Lag vs. missing structure.

Figure 3: **Fixed-clock editing stops structurally early.** Larger terminal lag is associated with more missing target structure.

Table 1: **User study.** Preference rates for prompt alignment, visual quality, and shape preservation. (Detailed in Appendix I)

Criterion	3DEdit Former	Nano3D	Pace3D
Prompt Align.	12.81%	9.69%	**77.50%**
Visual Quality	18.75%	26.25%	**55.00%**
Content Pres.	14.69%	21.25%	**64.06%**

the target may alter unchanged content. Pace3D therefore uses the pretrained SLat model to decide whether a shared feature should remain target-driven or be restored from the source.

Pace3D queries the pretrained SLat flow under source and target conditions at the same noisy state $\boldsymbol{z}_t$ and compares their predicted clean features:

$$\widehat{\boldsymbol{z}}^{\mathbf{c}}_{0,t} = (1-\sigma_{\min})\boldsymbol{z}_t - \sigma_t \boldsymbol{v}^{\mathrm{SLat}}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t, \boldsymbol{c}), \qquad \boldsymbol{d}_i(t) \quad = \widehat{\boldsymbol{z}}^{\mathrm{tgt}}_{0,t,i} - \widehat{\boldsymbol{z}}^{\mathrm{src}}_{0,t,i}. \tag{12}$$

Large source–target disagreement indicates target-driven content, while low disagreement favors source restoration. Pace3D normalizes disagreement as $r_i(t) = \|\boldsymbol{d}_i(t)\|_2/(\sigma_t \bar{v}_i(t))$, where $\bar{v}_i(t)$ is the mean norm of the two conditional velocities at token i. It averages $r_i(t)$ over high-noise steps, smooths it on the support grid, and maps it to a preservation weight $w_i \in [0,1]$ using a fixed threshold (Appendix A.5). After target-conditioned sampling, it applies geometry-aware fusion:

$$\boldsymbol{z}^{\mathrm{fuse}}_i = \begin{cases} w_i \boldsymbol{z}^{\mathrm{src}}_{j^\star(i)} + (1-w_i)\boldsymbol{z}^{\mathrm{tgt}}_i, & i \in \mathcal{I}, \\ \boldsymbol{z}^{\mathrm{tgt}}_i, & i \in \Omega_0 \setminus \mathcal{I}, \end{cases} \tag{13}$$

where $j^\star(i)$ is the source token at the same sparse-grid coordinate. Source restoration requires geometric correspondence, and each shared feature is restored in proportion to its preservation weight, which decreases with the aggregated relative disagreement. Instead of user masks, this weighting relies on model-internal evidence. Thus, Stage I allocates extra opportunity to unresolved structure, while Stage II modulates source restoration by cross-condition evidence.

4.4 End-to-End Inference

Algorithm 1 in Appendix A.6 summarizes the full procedure. Pace3D uses one auxiliary SS pass to construct the target proxy and decodes the reference path once to form $\mathcal{C}$. Stage I incurs decoder queries and reset-triggered SS flow evaluations capped by $R_{\max}$, while Stage II adds source-conditioned SLat probes at a few timesteps. No training or per-asset optimization is required. Since these operations occur at inference, the matched-compute comparison in Sec. 5, detailed in Appendix D.2, ensures that the gains reflect progress-aware allocation rather than more computation.

5 Experiments

Benchmark and implementation: We evaluate Pace3D on Eval3DEdit (Zhou et al., 2026), built on the pretrained hierarchical prior (Xiang et al., 2025); all baselines use their official defaults (Appendix B.1). **Metrics:** Following (Ye et al., 2025; Zhou et al., 2026), we measure source fidelity with PCPC, DINOsrc, and FID, and target alignment with PCImg, PCTxt, MVImg, MVTxt, and $\mathrm{DINO}_{\mathrm{tgt}}$. Results are averaged over three seeds; evaluation details are in Appendix B.2.

5.1 Does Fixed-Clock Editing Stop Structurally Early?

Firstly, we analyze 12 structural edits with paired source and ground-truth SS latents. Decoded states from a fixed-clock, training-free flow trajectory (Kulikov et al., 2025) are matched to their nearest phases on an oracle trajectory under the same schedule, defining structural lag; missing target structure is measured on the final support (Appendix C). Figure 3 shows that lag grows during sampling and persists at the endpoint, while larger terminal lag corresponds to more missing target structure, confirming that elapsed sampling time does not guarantee structural completion.

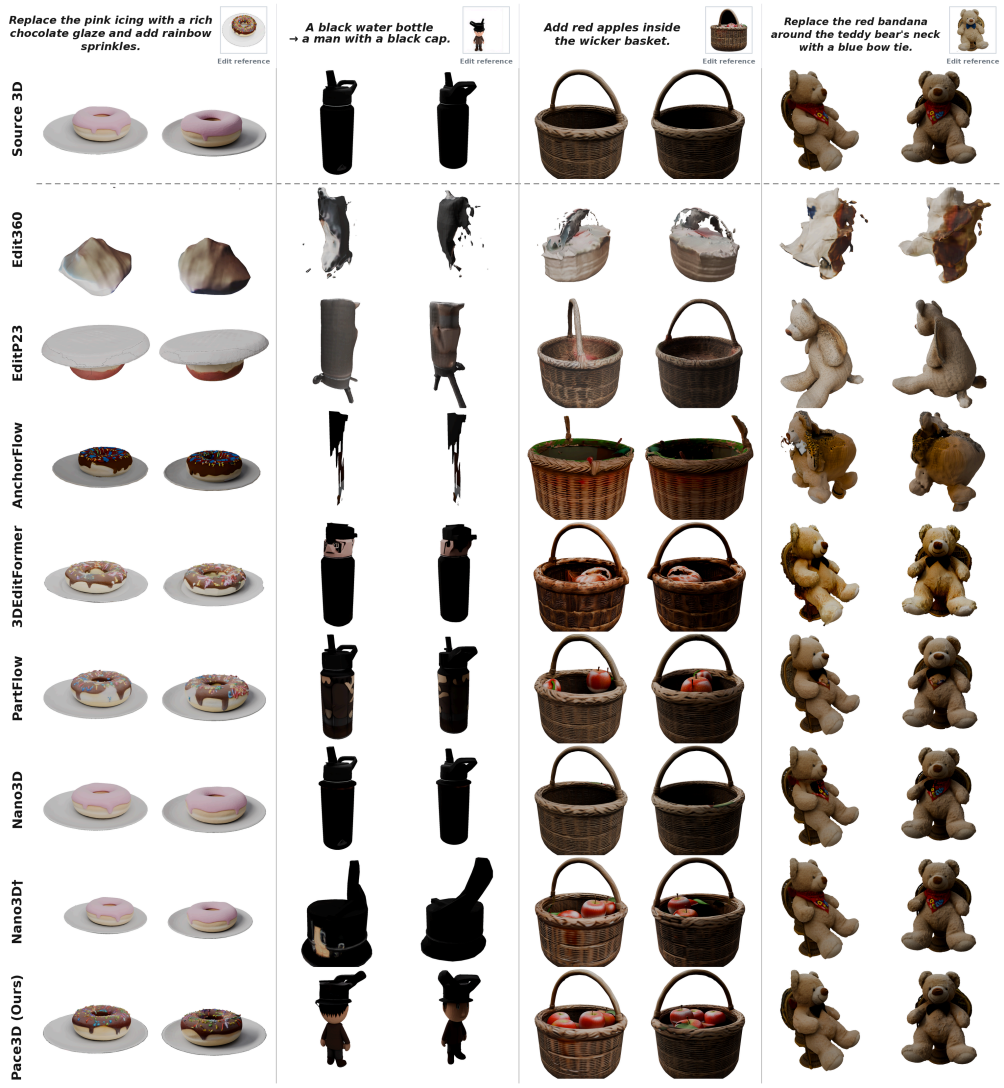


Figure 4: **Qualitative comparison.** † denotes the SS-NFE matched baseline.

5.2 Main Results

We compare Pace3D with Nano3D (Ye et al., 2025), AnchorFlow (Zhou et al., 2026), PartFlow (Weng et al., 2026), 3DEditFormer (Xia et al., 2025), EditP23 (Bar-On et al., 2025), and Edit360 (Huang et al., 2025). We additionally evaluate a Nano3D variant with its SS-stage NFE matched to Pace3D as an SS-NFE baseline. See Appendix B.1 for implementation details.

Qualitative Comparison. Figure 4 compares representative editing results. Edit360 and EditP23 often degrade geometry or fail to realize the target edit. Methods built on the hierarchical 3D prior preserve the source more reliably, but frequently under-edit or leave target structures incomplete. In contrast, Pace3D achieves more complete target realization while maintaining coherent source geometry, as seen in the bottle-to-person and apple-basket examples. The donut and teddy-bear cases further show that Pace3D can localize appearance changes without disturbing the underlying shape. More qualitative results are provided in Appendix F.

Quantitative Comparison. Table 2 shows that Pace3D achieves the best results on all five target-alignment metrics while maintaining strong source fidelity, including the second-best DINO_{src}. Increasing Nano3D's SS-NFE improves alignment but degrades source preservation, whereas Pace3D

Table 2: **Quantitative comparison on Eval3DEdit.** Best and second-best available results are shown in **bold** and underlined, respectively. † denotes the closest available SS-NFE Nano3D control.

		Source fidelity			Target alignment				
Method	**Base model**	PC_{PC} ↑	$DINO_{src}$ ↑	FID↓	PC_{Img} ↑	PC_{Txt} ↑	MV_{Img} ↑	MV_{Txt} ↑	$DINO_{tgt}$ ↑
Edit360	Multi-view diffusion	0.2707	0.6226	166.05	0.1878	0.1287	0.5290	0.3964	0.4226
EditP23	Multi-view diffusion	0.5498	0.7367	118.42	0.2870	0.1974	0.6276	0.4508	0.4922
AnchorFlow	Hierarchical 3D	0.6612	0.7489	70.53	0.3295	0.2324	0.6847	0.4712	0.5397
3DEditFormer	Hierarchical 3D	0.7988	0.7950	54.90	0.3578	0.2465	0.7098	0.4771	0.5620
PartFlow	Hierarchical 3D	0.8154	**0.8058**	51.68	0.3619	0.2473	0.7128	0.4776	0.5620
Nano3D	Hierarchical 3D	**0.8185**	0.7969	50.11	0.3545	0.2431	0.7083	0.4757	0.5604
Nano3D†	Hierarchical 3D	0.7653	0.7918	55.77	0.3559	0.2468	0.7190	0.4844	0.5698
Pace3D	Hierarchical 3D	0.7997	0.8009	**50.06**	**0.3653**	**0.2566**	**0.7291**	**0.4899**	**0.5733**

Table 3: **Component ablation.** Components are cumulatively added to the baseline.

Variant	$DINO_{src}$ ↑	PC_{PC} ↑	PC_{Txt} ↑	MV_{Img} ↑
Baseline	0.7957	0.7604	0.2409	0.7071
+ Controller	0.7877	0.7383	0.2532	0.7218
+ Src. Align.	0.7914	0.7815	0.2537	0.7229
+ CCC	**0.8009**	**0.7997**	**0.2566**	**0.7291**

Table 4: **Controller comparison.** Reset strategies under matched compute.

Method	$DINO_{src}$ ↑	PC_{PC} ↑	PC_{Txt} ↑	MV_{Img} ↑
Random	0.7950	0.7846	0.2438	0.7092
Fixed	0.7984	0.7782	0.2489	0.7191
Pace3D	**0.8009**	**0.7997**	**0.2566**	**0.7291**

outperforms this control across all metrics. The user study in Table 1 further shows a clear preference for Pace3D across prompt alignment, visual quality, and content preservation.

5.3 Ablation Study

Table 3 reports the contribution of each component in our framework. Starting from the baseline, introducing the *Controller* improves target alignment, while adding *Source Alignment* further enhances source preservation and perceptual quality. Finally, incorporating the proposed *Cross-Condition Consensus (CCC)* consistently improves all evaluation metrics and achieves the best overall performance. These results demonstrate that the three components play complementary roles, and that effectively coordinating source fidelity with target-conditioned editing is crucial for producing faithful and consistent results.

5.4 Efficiency, Robustness, and Limitations

Efficiency. Table 2 and Fig. 4 show that, under matched SS-NFE, Pace3D achieves stronger target alignment while maintaining source fidelity, demonstrating more effective compute allocation. Detailed compute-matched results are provided in Table 6 (Appendix D.2).

Controller robustness. Under matched editing steps, the progress-aware controller outperforms fixed and random reset policies (Table 4), confirming that the gains come from adaptive horizon allocation rather than additional steps alone. Further controller analyses are provided in Appendix D.

Limitations Discussion. Pace3D inherits reconstruction limitations from the underlying SLat backbone, including loss of fine geometry and texture that occurs before our controller is applied. See Appendix J for detailed failure cases.

6 Conclusion

In this paper, we identify structural progress–clock mismatch as a key termination failure in hierarchical training-free 3D editing, where the SS trajectory may terminate before sufficient structural support is formed. Pace3D addresses this issue with proxy-calibrated decoder feedback for adaptive horizon control, source alignment for stable updates, and temporal cross-condition consensus for selective restoration. Together, these designs improve structural completeness and stability across hierarchical stages. More broadly, this work suggests that when continuous generative trajectories determine discrete intermediate structure, decoded structure, rather than sampling time alone, should guide adaptive horizon allocation.

AI USE STATEMENT

In this work, we used generative AI tools for limited assistance with manuscript preparation and research workflow. For mathematical content, these tools were used only to check the presentation and notation of author-written mathematical expressions and proofs, such as symbol consistency and possible typographical omissions. They were not used to formulate mathematical claims, develop proof ideas, or write proofs.

We also used generative AI tools to improve manuscript readability, assist with creating and editing software code, and help organize experimental records. All AI-assisted outputs were reviewed and verified by the authors. In particular, suggestions related to mathematical content were independently checked, AI-assisted code was reviewed and tested for correctness, and manuscript edits and experimental records were verified against the original content and experimental logs.

The authors take full responsibility for the final content of this work, including its text, claims, code, proofs, experimental records, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work includes a user study in which participants provided ratings of generated results. Participation was voluntary, and participants were informed about the purpose and procedure of the study before participation. The study did not collect personally identifiable or sensitive information, and the collected ratings were analyzed and reported only in aggregate. The study involved minimal risk to participants, and the collected data were used solely for research purposes.

REPRODUCIBILITY STATEMENT

We support reproducibility by providing the mathematical assumptions and derivations underlying structural progress and remaining capacity in Appendix A.1, together with the phase-matching and clock-reset analysis in Appendix A.3. The complete inference procedure is specified in Appendix A.6, while the backbone, sampling schedules, guidance settings, controller parameters, and feature-preservation settings are documented in Appendix B.1. The benchmark, preprocessing, rendering setup, and metric definitions are described in Appendix B.2.

REFERENCES

Chengyu Bai, Jintao Chen, Xiang Bai, Yilong Chen, Qi She, Ming Lu, and Shanghang Zhang. Uniedit-i: Training-free image editing for unified VLM via iterative understanding, editing and verifying. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2026, Denver, CO, USA, June 3-7, 2026*, pp. 29750–29759, 2026.

Roi Bar-On, Dana Cohen-Bar, and Daniel Cohen-Or. Editp23: 3d editing via propagation of image prompts to multi-view. *arXiv preprint arXiv:2506.20652*, 2025.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 16273–16282, 2025.

Yuzhu Chen, Fengxiang He, Shi Fu, Xinmei Tian, and Dacheng Tao. Adaptive time-stepping schedules for diffusion models. In *Uncertainty in Artificial Intelligence, 15-19 July 2024, Universitat Pompeu Fabra, Barcelona, Spain*, volume 244, pp. 685–697, 2024.

Shaocong Dong, Lihe Ding, Zhanpeng Huang, Zibin Wang, Tianfan Xue, and Dan Xu. Interactive3d: Create what you want by interactive 3d generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 4999–5008, 2024.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 640–649, 2025.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6626–6637, 2017.

Junchao Huang, Xinting Hu, Shaoshuai Shi, Zhuotao Tian, and Li Jiang. Edit360: 2d image edits to 3d assets from any angle. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 16618–16628, 2025.

Jeongsol Kim, Yeobin Hong, and Jong Chul Ye. Flowalign: Trajectory-regularized, inversion-free flow-based image editing. *arXiv preprint arXiv:2505.23145*, 2025.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 19721–19730, 2025.

Zeqiang Lai, Yunfei Zhao, Haolin Liu, Zibo Zhao, Qingxiang Lin, Huiwen Shi, Xianghui Yang, Mingxin Yang, Shuhui Yang, Yifei Feng, Sheng Zhang, Xin Huang, Di Luo, Fan Yang, Fang Yang, Lifu Wang, Sicong Liu, Yixuan Tang, Yulin Cai, Zebin He, Tian Liu, Yuhong Liu, Jie Jiang, Linus, Jingwei Huang, and Chunchao Guo. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details. *arXiv preprint arXiv:2506.16504*, 2025.

Lin Li, Zehuan Huang, Haoran Feng, Gengxiong Zhuang, Rui Chen, Chunchao Guo, and Lu Sheng. Voxhammer: Training-free precise and coherent 3d editing in native 3d space. In *International Conference on 3D Vision, 3DV 2026, Vancouver, BC, Canada, March 20-23, 2026*, pp. 1281–1292, 2026.

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, SIGGRAPH Conference Papers 2025, Vancouver, BC, Canada, August 10-14, 2025*, pp. 81:1–81:10, 2025a.

Zixiang Li, Yue Song, Jianing Peng, Ting Liu, Jun Huang, Xiaochao Qu, Luoqi Liu, Wei Wang, Yao Zhao, and Yunchao Wei. On exact editing of flow-based diffusion models. *arXiv preprint arXiv:2512.24015*, 2025b.

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 300–309, 2023.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024, 2024.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

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.

Yurui Qian, Qi Cai, Yingwei Pan, Yehao Li, Ting Yao, Qibin Sun, and Tao Mei. Boosting diffusion models with moving average sampling in frequency domain. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 8911–8920, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139, pp. 8748–8763, 2021.

Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235, pp. 42947–42975, 2024.

Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 430–440, 2023.

Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 20902–20911, 2024.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Jiawei Weng, Saining Zhang, Zhenxin Diao, Peishuo Li, Henghaofan Zhang, Junhao Chen, and Hao Zhao. Feedforward 3d editing learns from semantic-part transformation. *arXiv preprint arXiv:2605.27351*, 2026.

Cunqi Wu, Peng Zhou, Jie Qin, and Qi Tian. 100editor: 100+ views per batch and minute-scale view-consistent 3d editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 8449–8460, June 2026.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3d: Scalable image-to-3d generation via 3d latent diffusion transformer. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Ruihao Xia, Yang Tang, and Pan Zhou. Towards scalable and consistent 3d editing. *arXiv preprint arXiv:2510.02994*, 2025.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 21469–21480, 2025.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14419–14429, June 2026.

Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi S. Jaakkola. Restart sampling for improving generative processes. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, 2023.

Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 22552–22562, 2023.

Junliang Ye, Shenghao Xie, Ruowen Zhao, Zhengyi Wang, Hongyu Yan, Wenqiang Zu, Lei Ma, and Jun Zhu. NANO3D: A training-free approach for efficient 3d editing without masks. *arXiv preprint arXiv:2510.15019*, 2025.

Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4):92:1–92:16, 2023.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Trans. Graph.*, 43(4):120:1–120:20, 2024.

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, Huiwen Shi, Sicong Liu, Junta Wu, Yihang Lian, Fan Yang, Ruining Tang, Zebin He, Xinzhou Wang, Jian Liu, Xuhui Zuo, Zhuo Chen, Biwen Lei, Haohan Weng, Jing Xu, Yiling Zhu, Xinhai Liu, Lixin Xu, Changrong Hu, Tianyu Huang, Lifu Wang, Jihong Zhang, Meng Chen, Liang Dong, Yiwen Jia, Yulin Cai, Jiaao Yu, Yixuan Tang, Hao Zhang, Zheng Ye, Peng He, Runzhou Wu, Chao Zhang, Yonghao Tan, Jie Xiao, Yangyu Tao, Jianchen Zhu, Jinbao Xue, Kai Liu, Chongqing Zhao, Xinming Wu, Zhichao Hu, Lei Qin, Jianbing Peng, Zhan Li, Minghui Chen, Xipeng Zhang, Lin Niu, Paige Wang, Yingkai Wang, Haozhao Kuang, Zhongyi Fan, Xu Zheng, Weihao Zhuang, YingPing He, Tian Liu, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, Jingwei Huang, and Chunchao Guo. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Zhenglin Zhou, Fan Ma, Chengzhuo Gui, Xiaobo Xia, Hehe Fan, Yi Yang, and Tat-Seng Chua. Anchorflow: Training-free 3d editing via latent anchor-aligned flows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14387–14397, 2026.

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 Trans. Graph.*, 43(4): 121:1–121:12, 2024.

A Analysis and Method Details

We provide additional technical details and analyses of our method. Specifically, we present the exact training-free FlowEdit (Kulikov et al., 2025) queries and derive the progress–capacity formulation, describe the end-to-end sampling procedure, analyze decoder-space phase matching and clock rewinding, examine the stabilizing role of Source Alignment, and detail the temporal aggregation strategy employed in Cross-Condition Consensus.

A.1 Flow-Editing Setup and Progress–Capacity Details

We begin by specifying the flow-editing field in Eq. (1) and deriving a time-varying extension of the remaining-horizon bound in Eq. (6). Let $\alpha_t = 1 - t$ and $\sigma_t = \sigma_{\min} + (1 - \sigma_{\min})t$. Given shared noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, the source and target queries for training-free SS editing are defined as

$$\begin{aligned} \boldsymbol{x}_t^{\mathrm{src}} &= \alpha_t \boldsymbol{x}^{\mathrm{src}} + \sigma_t \boldsymbol{\epsilon}, \\ \boldsymbol{x}_t^{\mathrm{tgt}} &= \boldsymbol{x}_t + \boldsymbol{x}_t^{\mathrm{src}} - \boldsymbol{x}^{\mathrm{src}}, \\ \boldsymbol{f}_t(\boldsymbol{x}_t) &= \mathbb{E}_{\boldsymbol{\epsilon}}\left[\boldsymbol{v}_{\boldsymbol{\theta}}^{\mathrm{SS}}(\boldsymbol{x}_t^{\mathrm{tgt}}, t, \boldsymbol{c}_{\mathrm{tgt}}) - \boldsymbol{v}_{\boldsymbol{\theta}}^{\mathrm{SS}}(\boldsymbol{x}_t^{\mathrm{src}}, t, \boldsymbol{c}_{\mathrm{src}})\right]. \end{aligned} \tag{14}$$

This field governs latent motion through Eq. (1). To relate that motion to structural completion, consider the signed target margin g_j and debt m_j defined in Eq. (3). Along the editing trajectory, the chain rule and Eq. (4) give

$$\frac{\mathrm{d}}{\mathrm{d}\tau} g_j(\boldsymbol{x}_\tau) = \nabla_{\boldsymbol{x}} g_j(\boldsymbol{x}_\tau)^\top \boldsymbol{f}_\tau(\boldsymbol{x}_\tau) = -q_j(\tau). \tag{15}$$

Integrating from the current time t to 0 yields

$$g_j(\boldsymbol{x}_0) = g_j(\boldsymbol{x}_t) + \int_0^t q_j(\tau)\,\mathrm{d}\tau. \tag{16}$$

For an unfinished event, $g_j(\boldsymbol{x}_t) = -m_j(\boldsymbol{x}_t)$, so the remaining integral must cover the current debt for the endpoint margin to be nonnegative. This recovers Eq. (5) and identifies the quantity that the remaining-horizon bound must control.

To bound this integral while allowing decoder sensitivity and field magnitude to vary with time, suppose that, along the remaining trajectory on $[0, t]$, $\|\nabla_{\boldsymbol{x}} g_j(\boldsymbol{x}_\tau)\|_2 \le L_j(\tau)$ and $\|\boldsymbol{f}_\tau(\boldsymbol{x}_\tau)\|_2 \le B(\tau)$, where L_j and B are nonnegative and their product is integrable. By Cauchy–Schwarz,

$$[q_j(\tau)]_+ \le |q_j(\tau)| \le \|\nabla_{\boldsymbol{x}} g_j(\boldsymbol{x}_\tau)\|_2 \|\boldsymbol{f}_\tau(\boldsymbol{x}_\tau)\|_2 \le L_j(\tau) B(\tau). \tag{17}$$

Integrating gives the time-varying upper bound on remaining structural capacity,

$$\int_0^t q_j(\tau)\,\mathrm{d}\tau \le \int_0^t [q_j(\tau)]_+\,\mathrm{d}\tau \le C_j^{\mathrm{ub}}(t) := \int_0^t L_j(\tau) B(\tau)\,\mathrm{d}\tau. \tag{18}$$

The positive part counts all target-directed progress without subtracting any regress, making $C_j^{\mathrm{ub}}(t)$ an optimistic upper bound on the net margin gain. Substituting Eq. (18) into Eq. (16) gives, for an unfinished event,

$$g_j(\boldsymbol{x}_0) \le -m_j(\boldsymbol{x}_t) + C_j^{\mathrm{ub}}(t). \tag{19}$$

Thus the completion condition in Eq. (5) cannot be met when the debt exceeds this capacity bound:

$$m_j(\boldsymbol{x}_t) > C_j^{\mathrm{ub}}(t) \quad \Longrightarrow \quad g_j(\boldsymbol{x}_0) < 0. \tag{20}$$

The certificate is deliberately one-sided. A small upper bound can prove that an event cannot finish, but a large upper bound does not guarantee completion, because the learned field may fail to use that capacity in the target-directed decoder direction.

The main-text bound is the special case with uniform bounds $L_j(\tau) \le \bar{L}_j$ and $B(\tau) \le \bar{B}$ over the remaining interval. Then Eq. (18) reduces to $C_j^{\mathrm{ub}}(t) \le \bar{L}_j \bar{B}\, t$, and Eq. (20) gives exactly the implication in Eq. (6). With finite uniform bounds near $t = 0$, this capacity bound vanishes as $t \to 0$, whereas the state-dependent debt need not. The loss of remaining capacity therefore follows from the shrinking integration horizon; it does not require the field magnitude itself to vanish.

Late-time refinement and unresolved structural debt. Beyond the shrinking horizon established above, late-time editing may also favor local refinement of the current structure, limiting the useful progress available to clear unresolved structural debt. To characterize this late-time regime, we express each branch in Eq. (14) through its clean-latent prediction,

$$\widehat{\boldsymbol{x}}_{0,t}^{b} = (1 - \sigma_{\min})\boldsymbol{x}_t^b - \sigma_t \boldsymbol{v}_{\boldsymbol{\theta}}^{\mathrm{SS}}(\boldsymbol{x}_t^b, t, \boldsymbol{c}_b), \qquad b \in \{\mathrm{src}, \mathrm{tgt}\}. \tag{21}$$

Let $\boldsymbol{e}_t^{\mathrm{src}} = \widehat{\boldsymbol{x}}_{0,t}^{\mathrm{src}} - \boldsymbol{x}^{\mathrm{src}}$ be the source reconstruction error. Subtracting the two predictions and using $\boldsymbol{x}_t^{\mathrm{tgt}} - \boldsymbol{x}_t^{\mathrm{src}} = \boldsymbol{x}_t - \boldsymbol{x}^{\mathrm{src}}$ gives the exact identity

$$-\boldsymbol{f}_t(\boldsymbol{x}_t) = \frac{\mathbb{E}_{\boldsymbol{\epsilon}}[\widehat{\boldsymbol{x}}_{0,t}^{\mathrm{tgt}}] - \boldsymbol{x}_t - \mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{e}_t^{\mathrm{src}}] + \sigma_{\min}(\boldsymbol{x}_t - \boldsymbol{x}^{\mathrm{src}})}{\sigma_t}. \tag{22}$$

When the source reconstruction error and noise-floor correction are negligible relative to the target prediction residual, this simplifies to

$$-\boldsymbol{f}_t(\boldsymbol{x}_t) \approx \frac{\mathbb{E}_{\boldsymbol{\epsilon}}[\widehat{\boldsymbol{x}}_{0,t}^{\mathrm{tgt}}] - \boldsymbol{x}_t}{\sigma_t}. \tag{23}$$

Since sampling proceeds toward decreasing t, Eq. (23) shows that the update pulls the editable state toward the target-conditioned clean prediction. Crucially, this prediction is inferred from the current query x_t and is therefore anchored to the current state, rather than being a fixed target latent specified independently of the editing trajectory. The query itself approaches the editable state at low noise:

$$\boldsymbol{x}_t^{\mathrm{tgt}} - \boldsymbol{x}_t = -t\boldsymbol{x}^{\mathrm{src}} + \sigma_t\boldsymbol{\epsilon} \approx t(\boldsymbol{\epsilon} - \boldsymbol{x}^{\mathrm{src}}), \qquad \sigma_{\min} \le t \ll 1. \tag{24}$$

Thus the target branch receives an increasingly small perturbation of the current editable state, including any structural omissions it still contains. Prior studies of image diffusion models (Rissanen et al., 2023; Yang et al., 2023; Qian et al., 2024) have consistently revealed a coarse-to-fine temporal evolution, with related behavior also observed in flow-based image editing (Kulikov et al., 2025). Global and low-frequency structure is established predominantly at earlier stages, whereas local texture and high-frequency details are progressively refined later in the trajectory. This temporal bias provides a complementary explanation for structural lag. When the target prediction is evaluated late in the trajectory, it may primarily refine the structure already encoded in the current editable state, while providing limited progress toward a support event that remains structurally absent. By Eq. (4), only the component of this correction along $\nabla_{\boldsymbol{x}} g_j$ contributes to reducing the corresponding debt.

As t approaches zero, the low-noise query together with late-time conditioning may cause the target branch to treat the current editable state as being close to completion, even when unresolved structural debt remains. In terms of Eq. (4), this mismatch implies that the remaining target-directed progress $q_j(t)$ may be insufficient to discharge the outstanding debt before termination. The diffusion clock therefore enters a late refinement regime before the editable state has completed the required structural changes, while the diminishing horizon in Eq. (18) further limits the remaining capacity to correct this discrepancy.

A.2 Properties of Discrete Structural Phase Matching

In this subsection, we first derive the local debt–phase relation in Eq. (8) and then explain the decoder-space phase matching in Eq. (9).

Local debt–phase relation. Let $\ell_j(t) = D_j(P_t)$ be locally monotone around a threshold crossing $\ell_j(\tau_j) = \kappa$. Define $\widehat{s}_j = 2O_j(P_0) - 1$ and $\widehat{m}_j^{\mathrm{ref}}(P_t) = [-\widehat{s}_j(\ell_j(t) - \kappa)]_+$. Taylor expansion around τ_j gives

$$\widehat{m}_j^{\mathrm{ref}}(P_t) = \left|\dot{\ell}_j(\tau_j)\right|[t - \tau_j]_+ + o(|t - \tau_j|)\,. \tag{25}$$

Thus, close to a reference occupancy event, residual decoder debt can be represented by a phase gap up to a local scale factor.

Why not project the full latent? We analyze full-latent projection to explain the choice of decoder-space matching in Eq. (9). The source-to-proxy path in Eq. (7) can equivalently be parameterized by $P(\alpha) = \boldsymbol{x}^{\mathrm{src}} + \alpha\widehat{\boldsymbol{\Delta}}$, where $\widehat{\boldsymbol{\Delta}} = \boldsymbol{x}^{\mathrm{proxy}} - \boldsymbol{x}^{\mathrm{src}}$ and $\alpha = 1 - t/t_{\mathrm{start}}$. A direct latent-space phase estimate is

$$\widehat{\alpha}_t = \arg\min_{\alpha} \frac{1}{2}\left\|\boldsymbol{x}_t - P(\alpha)\right\|_2^2 = \frac{\left\langle \boldsymbol{x}_t - \boldsymbol{x}^{\mathrm{src}}, \widehat{\boldsymbol{\Delta}} \right\rangle}{\left\|\widehat{\boldsymbol{\Delta}}\right\|_2^2}. \tag{26}$$

Write $\boldsymbol{x}_t = P(\alpha^\star) + \boldsymbol{e}$, where $\boldsymbol{e}$ is the off-path displacement. Assuming $\widehat{\boldsymbol{\Delta}} \neq \boldsymbol{0}$, the phase error satisfies

$$|\widehat{\alpha}_t - \alpha^\star| = \frac{\left|\left\langle \boldsymbol{e}, \widehat{\boldsymbol{\Delta}} \right\rangle\right|}{\left\|\widehat{\boldsymbol{\Delta}}\right\|_2^2} \le \frac{\|\boldsymbol{e}\|_2}{\left\|\widehat{\boldsymbol{\Delta}}\right\|_2}. \tag{27}$$

This expression makes two failure modes explicit. A small source-to-proxy displacement amplifies phase error, and unrelated variation parallel to the proxy direction is misread as progress. Decoder-space matching avoids both problems whenever those variations do not change occupancy.

Why reference-constant voxels do not shift the phase. Let $\mathcal{J}_{\text{vary}}$ denote the set of voxels whose occupancy changes across the cached reference path,

$$\mathcal{J}_{\text{vary}} = \{j : \exists\, k, \ell,\ O_j(P_{t_k}) \neq O_j(P_{t_\ell})\}.$$

For every $j \notin \mathcal{J}_{\text{vary}}$, the reference occupancy $O_j(P_{t_k})$ is constant over k. Therefore,

$$\|O(\boldsymbol{x}_t) \oplus O(P_{t_k})\|_1 = \sum_{j \in \mathcal{J}_{\text{vary}}} \mathbf{1}[O_j(\boldsymbol{x}_t) \neq O_j(P_{t_k})] + C(\boldsymbol{x}_t), \tag{28}$$

where $C(\boldsymbol{x}_t)$ is independent of k. Thus, reference-constant voxels can shift all Hamming distances by the same amount but cannot change the minimizing cached phase. If adjacent cache entries have identical occupancy, they are indistinguishable under this objective; we choose the larger timestep to conservatively allocate more remaining horizon.

A.3 Effect of Sampling-Time Reset on Structural Completion

Suppose the current clock is t and Hamming matching returns $\tau = t_{\widehat{k}} > t$. Pace3D resets the sampling time while preserving the latent,

$$\widetilde{\boldsymbol{x}}_\tau \leftarrow \boldsymbol{x}_t. \tag{29}$$

Consequently, every decoder margin and debt value is unchanged at the reset:

$$g_j(\widetilde{\boldsymbol{x}}_\tau) = g_j(\boldsymbol{x}_t), \qquad m_j(\widetilde{\boldsymbol{x}}_\tau) = m_j(\boldsymbol{x}_t). \tag{30}$$

Let $\widetilde{q}_j$ denote the structural progress rate along the trajectory after the reset. Applying Eq. (5) to the extended horizon shows that event j completes if

$$\int_0^\tau \widetilde{q}_j(s)\,\mathrm{d}s \geq m_j(\boldsymbol{x}_t). \tag{31}$$

Without a reset, the corresponding integral is limited to $[0, t]$. The reset therefore provides a longer interval over which progress may accumulate, but it does not alter the current debt or guarantee that the field points in a useful direction. Rebuilding both queries at the larger noise level may also ease the late-time refinement bias discussed in Appendix A.1, giving the model more scope to revise the incomplete structure.

Recomputing the editing field. After resetting the clock to τ, we reconstruct the source and target queries using the retained editable state and a shared noise sample $\boldsymbol{\epsilon}$:

$$\boldsymbol{x}_\tau^{\boldsymbol{c}_{\text{src}}} = \alpha_\tau \boldsymbol{x}^{\text{src}} + \sigma_\tau \boldsymbol{\epsilon}, \qquad \boldsymbol{x}_\tau^{\boldsymbol{c}_{\text{tgt}}} = \widetilde{\boldsymbol{x}}_\tau + \boldsymbol{x}_\tau^{\boldsymbol{c}_{\text{src}}} - \boldsymbol{x}^{\text{src}}. \tag{32}$$

Both conditional velocities are evaluated at τ, and their difference defines the source-to-target editing field in Eq. (14) at the reset state.

Because the proxy is approximate and the edited trajectory need not follow the reference path exactly, the controller may reset more than once. A maximum reset count bounds the added computation. These safeguards affect only when the controller can act; they do not change the phase-matching objective.

A.4 A Stability Bound for Source Alignment

We show how the update in Eq. (11) attenuates transient deviations while allowing persistent editing signals to contribute, then bound the accumulation of sustained off-target updates. Let $\boldsymbol{\Delta}_n = \boldsymbol{x}_{t_n} - \boldsymbol{x}^{\text{src}}$ and $\boldsymbol{u}_n = (t_{n+1} - t_n)\boldsymbol{f}_{t_n}(\boldsymbol{x}_{t_n})$, with $0 < \eta < 1$. The update can be written as

$$\boldsymbol{\Delta}_{n+1} = (1-\eta)\boldsymbol{\Delta}_n + \boldsymbol{u}_n, \qquad \boldsymbol{\Delta}_N = \sum_{n=0}^{N-1} (1-\eta)^{N-1-n} \boldsymbol{u}_n, \tag{33}$$

where $\boldsymbol{\Delta}_0 = \mathbf{0}$ because editing starts from the source. Each past update is therefore attenuated by $1 - \eta$ at every subsequent step.

For analysis, decompose the updates along the editing trajectory as $\boldsymbol{u}_n = \boldsymbol{u}_n^{\text{edit}} + \boldsymbol{u}_n^{\text{off}}$. The accumulated off-target component is

$$\boldsymbol{\Delta}_N^{\text{off}} = \sum_{n=0}^{N-1} (1-\eta)^{N-1-n} \boldsymbol{u}_n^{\text{off}}. \tag{34}$$

Suppose an off-target disturbance ends at step n_0, so that $\boldsymbol{u}_n^{\text{off}} = \mathbf{0}$ for $n \geq n_0$. Equation (34) then gives, for $N \geq n_0$,

$$\boldsymbol{\Delta}_N^{\text{off}} = (1-\eta)^{N-n_0} \boldsymbol{\Delta}_{n_0}^{\text{off}}, \qquad \left\|\boldsymbol{\Delta}_N^{\text{off}}\right\|_2 = (1-\eta)^{N-n_0} \left\|\boldsymbol{\Delta}_{n_0}^{\text{off}}\right\|_2. \tag{35}$$

Thus a transient deviation decays geometrically after its input ceases. In contrast, an editing direction consistently supplied by the target field receives new contributions at successive steps in Eq. (33). This is the temporal-persistence mechanism described in Sec. 4.2: continued field input can sustain an edit, while an unsupported deviation fades.

When off-target components are introduced repeatedly, Source Alignment prevents their effects from accumulating without attenuation by progressively decaying earlier contributions. Suppose $|\boldsymbol{u}_n^{\text{off}}|_2 \leq \beta$ at every step. The triangle inequality then gives

$$\left|\boldsymbol{\Delta}_N^{\text{off}}\right|_2 \leq \beta \sum_{k=0}^{N-1} (1-\eta)^k = \beta \frac{1-(1-\eta)^N}{\eta}. \tag{36}$$

Here, β bounds the magnitude of each newly introduced off-target increment. Without Source Alignment, the corresponding worst-case accumulation grows linearly as $N\beta$. In contrast, Eq. (36) is strictly smaller for $N > 1$ and converges to β/η as $N \to \infty$. Source Alignment therefore converts unattenuated accumulation into a geometrically discounted process, keeping the effect of persistent but bounded off-target leakage finite over an increasing number of updates.

The contraction applies uniformly to the accumulated displacement. Directions that remain supported by subsequent editing updates are continually reinforced, while unsupported deviations gradually decay. We therefore use a small η to preserve editing progress while damping stale or transient deviations.

A.5 Cross-Condition Consensus Details

We describe the relative conditional response, its aggregation over the high-noise interval, and the construction of preservation weights for the final feature fusion.

Conditional response. At the same noisy latent $\boldsymbol{z}_t$, define the difference between the velocities predicted under the target and source conditions as

$$\boldsymbol{\Delta}_v(t) = \boldsymbol{v}_{\boldsymbol{\theta}}^{\text{SLat}}(\boldsymbol{z}_t, t, \boldsymbol{c}_{\text{tgt}}) - \boldsymbol{v}_{\boldsymbol{\theta}}^{\text{SLat}}(\boldsymbol{z}_t, t, \boldsymbol{c}_{\text{src}}). \tag{37}$$

Both responses are evaluated without classifier-free guidance; guidance is applied separately to the target sampling update. For the linear noise schedule, the corresponding conditional score difference is

$$\boldsymbol{\Delta}_{\text{score}}(t) = \nabla_{\boldsymbol{z}_t} \log \frac{p_t(\boldsymbol{z}_t \mid \boldsymbol{c}_{\text{tgt}})}{p_t(\boldsymbol{z}_t \mid \boldsymbol{c}_{\text{src}})} = -\frac{\alpha_t}{\sigma_t} \boldsymbol{\Delta}_v(t). \tag{38}$$

The clean-feature difference in Eq. (12) instead satisfies

$$\boldsymbol{d}_i(t) = -\sigma_t [\boldsymbol{\Delta}_v(t)]_i = \frac{\sigma_t^2}{\alpha_t} [\boldsymbol{\Delta}_{\text{score}}(t)]_i, \qquad 0 < t < 1. \tag{39}$$

This identity relates velocity, score, and clean-feature comparisons. The relative response used in Sec. 4.3 is

$$r_i(t) = \frac{\|[\boldsymbol{\Delta}_v(t)]_i\|_2}{\bar{v}_i(t) + \varepsilon}, \qquad \bar{v}_i(t) = \tfrac{1}{2} \sum_{\boldsymbol{c} \in \{\boldsymbol{c}_{\text{src}}, \boldsymbol{c}_{\text{tgt}}\}} \left\|[\boldsymbol{v}_{\boldsymbol{\theta}}^{\text{SLat}}(\boldsymbol{z}_t, t, \boldsymbol{c})]_i\right\|_2, \tag{40}$$

with $\varepsilon = 10^{-6}$; up to ε, this equals $\|\boldsymbol{d}_i(t)\|_2 / (\sigma_t \bar{v}_i(t))$ in Sec. 4.3. By Eq. (12), each conditional prediction moves the current latent by $\widehat{\boldsymbol{z}}_{0,t}^{\boldsymbol{c}} - (1-\sigma_{\min})\boldsymbol{z}_t = -\sigma_t \boldsymbol{v}_{\boldsymbol{\theta}}^{\text{SLat}}(\boldsymbol{z}_t, t, \boldsymbol{c})$, so $\sigma_t \bar{v}_i(t)$ is the

mean length of this predicted displacement at token i. The response thus compares the gap between the two predicted clean features with how far each prediction moves from $\boldsymbol{z}_t$. Both quantities scale with σ_t. At higher noise, the one-step prediction extrapolates farther, and the clean-feature gap grows even when the two conditions disagree to the same relative extent. Dividing by $\sigma_t \bar{v}_i(t)$ removes this noise-level scaling, so $r_i(t)$ reflects only the relative disagreement between conditions.

Aggregating responses at high noise. Let $\mathcal{H} = \{k : t_k > t_{\mathrm{hi}}\}$ denote the selected sampling steps, and let $\Delta t_k = t_k - t_{k+1} > 0$ denote the timestep length. We compute a scalar response for each token by averaging the relative response over this interval:

$$R_i = \frac{\sum_{k\in\mathcal{H}} \Delta t_k \, r_i(t_k)}{\sum_{k\in\mathcal{H}} \Delta t_k}. \tag{41}$$

The weights account for unequal timestep lengths after schedule rescaling. Taking the relative magnitude at each step before averaging retains conditional response strength even when its direction changes across steps. This score uses the uncentered velocity differences within the selected window.

Spatial smoothing. We smooth R_i on the 64^3 target support to reduce isolated spatial fluctuations. Let M be its binary occupancy grid and let R^{grid} contain R_i at occupied coordinates and zero elsewhere. For a Gaussian smoothing operator $\mathcal{G}_s$ with standard deviation $s = 7$ voxels, define

$$E_i = \left[\frac{\mathcal{G}_s(R^{\mathrm{grid}})}{\max\{\mathcal{G}_s(M), \epsilon_{\mathrm{occ}}\}}\right]_i, \qquad \epsilon_{\mathrm{occ}} = 10^{-6}. \tag{42}$$

The operations in the ratio are pointwise, and $[\cdot]_i$ samples the grid at token i. Smoothing both the scores and occupancy prevents empty voxels from lowering scores near support boundaries. The implementation uses separable Gaussian kernels truncated at three standard deviations and replicated grid padding.

From edit evidence to preservation weights. For each shared token $i \in \mathcal{I}$, we set $z_i = E_i$ and map it to a preservation weight using a release threshold τ and a linear transition of half-width h:

$$w_i = \min\Big\{1, \max\Big\{0, \tfrac{1}{2} - \tfrac{z_i - \tau}{2h}\Big\}\Big\}. \tag{43}$$

Tokens with $z_i \geq \tau + h$ are treated as edited and keep their target features ($w_i = 0$), tokens with $z_i \leq \tau - h$ are candidates for source preservation ($w_i = 1$), and only tokens within h of the threshold are blended; $h \to 0$ recovers the hard threshold $w_i = 1 - \mathbf{1}[z_i > \tau]$. Because z_i is dimensionless, a single pair (τ, h) applies across assets, and the amount of released content follows the response magnitude of each asset.

Correspondence-aware feature fusion. The preservation weights are defined over shared coordinates and applied without spatial dilation. For feature fusion, source features are restored only at locations with exact source–target correspondence. Let $j^\star(i)$ denote the source token corresponding to target token i at the same sparse-grid coordinate. The final feature fusion is

$$\boldsymbol{z}_i^{\mathrm{fuse}} = \begin{cases} w_i \boldsymbol{z}^{\mathrm{src}}_{j^\star(i)} + (1 - w_i)\boldsymbol{z}_i^{\mathrm{tgt}}, & i \in \mathcal{I}, \\ \boldsymbol{z}_i^{\mathrm{tgt}}, & i \notin \mathcal{I}. \end{cases} \tag{44}$$

Thus, source features are restored only at shared coordinates, weighted by the preservation weights, while newly introduced tokens retain their target-generated features.

A.6 End-to-End Sampling Procedure

Algorithm 1 presents the full Pace3D editing procedure. It first constructs a target proxy to serve as a reference for structural progress. Stage I edits the sparse structure and uses decoder feedback to adjust the sampling time. Stage II generates features on the resulting support and selectively blends in source features to preserve unchanged regions.

Algorithm 1 Pace3D Sampling

Input: Source SS and SLat latents $\boldsymbol{x}^{\mathrm{src}}, \boldsymbol{z}^{\mathrm{src}}$; conditions $\boldsymbol{c}_{\mathrm{src}}, \boldsymbol{c}_{\mathrm{tgt}}$; pretrained SS and SLat flows $\boldsymbol{v}^{\mathrm{SS}}, \boldsymbol{v}^{\mathrm{SLat}}$; SS decoder O; time grids $\mathcal{T}_{\mathrm{SS}}, \mathcal{T}_{\mathrm{SLat}}$; controls $\theta, \eta, R_{\max}, t_{\mathrm{hi}}, \tau, h, \varepsilon$
Output: Edited 3D asset $\mathcal{Y}^{\mathrm{edit}}$

SS reference construction
$\boldsymbol{x}^{\mathrm{proxy}} \leftarrow \textsc{TargetProxy}(\boldsymbol{x}^{\mathrm{src}}, \boldsymbol{c}_{\mathrm{tgt}})$
$\mathcal{C} \leftarrow \{(t_k, O(P_{t_k}))\}_{k=0}^{K}$
$\boldsymbol{x} \leftarrow \boldsymbol{x}^{\mathrm{src}}$; $\Omega_{\mathrm{src}} \leftarrow \textsc{Coordinates}(\boldsymbol{z}^{\mathrm{src}})$; $i \leftarrow i_{\mathrm{start}}$; $n_{\mathrm{reset}} \leftarrow 0$

Stage I: structure editing with clock control
while $i < |\mathcal{T}_{\mathrm{SS}}| - 1$ **do**
  $(t, t^-) \leftarrow (\mathcal{T}_{\mathrm{SS}}[i], \mathcal{T}_{\mathrm{SS}}[i+1])$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $\boldsymbol{x}_t^{\mathrm{src}} \leftarrow \alpha_t \boldsymbol{x}^{\mathrm{src}} + \sigma_t \boldsymbol{\epsilon}$; $\boldsymbol{x}_t^{\mathrm{tgt}} \leftarrow \boldsymbol{x} + \boldsymbol{x}_t^{\mathrm{src}} - \boldsymbol{x}^{\mathrm{src}}$
  $\boldsymbol{f}_t \leftarrow \boldsymbol{v}^{\mathrm{SS}}(\boldsymbol{x}_t^{\mathrm{tgt}}, t, \boldsymbol{c}_{\mathrm{tgt}}) - \boldsymbol{v}^{\mathrm{SS}}(\boldsymbol{x}_t^{\mathrm{src}}, t, \boldsymbol{c}_{\mathrm{src}})$
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + (t^- - t)\boldsymbol{f}_t + \eta(\boldsymbol{x}^{\mathrm{src}} - \boldsymbol{x})$; $i \leftarrow i + 1$
  if the controller is active and $n_{\mathrm{reset}} < R_{\max}$ **then**
    $\widehat{k} \leftarrow \arg\min_k \|O(\boldsymbol{x}) \oplus O_k^{\mathrm{ref}}\|_1$
    if $t_{\widehat{k}} - \mathcal{T}_{\mathrm{SS}}[i] > \theta$ **then**
      $i \leftarrow \textsc{Index}(t_{\widehat{k}})$; $n_{\mathrm{reset}} \leftarrow n_{\mathrm{reset}} + 1$ ▷ reset sampling time, retain $\boldsymbol{x}$
    end if
  end if
end while
$\Omega_0 \leftarrow \{q : O_q(\boldsymbol{x}) = 1\}$; $\mathcal{I} \leftarrow \Omega_0 \cap \Omega_{\mathrm{src}}$
For $i \in \mathcal{I}$, let $j^\star(i)$ be the source token at the same coordinate

Stage II: cross-condition consensus and feature fusion
Initialize $\boldsymbol{z}_{t_0} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ on Ω_0; $R_i \leftarrow 0$ for $i \in \Omega_0$; $W \leftarrow 0$
for $(t, t^-) \in \mathcal{T}_{\mathrm{SLat}}$ **do**
  $\boldsymbol{v}_t^{\mathrm{tgt}} \leftarrow \boldsymbol{v}^{\mathrm{SLat}}(\boldsymbol{z}_t, t, \boldsymbol{c}_{\mathrm{tgt}})$
  if $t > t_{\mathrm{hi}}$ **then**
    $\boldsymbol{v}_t^{\mathrm{src}} \leftarrow \boldsymbol{v}^{\mathrm{SLat}}(\boldsymbol{z}_t, t, \boldsymbol{c}_{\mathrm{src}})$
    $r_i \leftarrow \|(\boldsymbol{v}_t^{\mathrm{tgt}} - \boldsymbol{v}_t^{\mathrm{src}})_i\|_2 / \big(\frac{1}{2}\|(\boldsymbol{v}_t^{\mathrm{tgt}})_i\|_2 + \frac{1}{2}\|(\boldsymbol{v}_t^{\mathrm{src}})_i\|_2 + \varepsilon\big)$ ▷ relative response
    $R_i \leftarrow R_i + (t - t^-)\, r_i$; $W \leftarrow W + (t - t^-)$
  end if
  $\boldsymbol{v}_t^{\mathrm{CFG}} \leftarrow \textsc{GuidedVelocity}(\boldsymbol{z}_t, t, \boldsymbol{c}_{\mathrm{tgt}})$
  $\boldsymbol{z}_{t^-} \leftarrow \boldsymbol{z}_t + (t^- - t)\boldsymbol{v}_t^{\mathrm{CFG}}$
end for
$\boldsymbol{z}^{\mathrm{tgt}} \leftarrow \boldsymbol{z}_{t_{\mathrm{end}}}$; $\boldsymbol{z}^{\mathrm{fuse}} \leftarrow \boldsymbol{z}^{\mathrm{tgt}}$; $R_i \leftarrow R_i / W$
Smooth R_i on Ω_0 to obtain z_i (Eq. (42))
$w_i \leftarrow \min\{1, \max\{0, \frac{1}{2} - \frac{z_i - \tau}{2h}\}\}$ for $i \in \mathcal{I}$ (Eq. (43))
$\boldsymbol{z}_i^{\mathrm{fuse}} \leftarrow w_i \boldsymbol{z}_{j^\star(i)}^{\mathrm{src}} + (1 - w_i)\boldsymbol{z}_i^{\mathrm{tgt}}$ for $i \in \mathcal{I}$

return $\mathcal{Y}^{\mathrm{edit}} \leftarrow \textsc{Decode}(\boldsymbol{z}^{\mathrm{fuse}}, \Omega_0)$

Target proxy construction. We obtain $\boldsymbol{x}^{\mathrm{proxy}}$ with an SDEdit-style pass in the SS latent space. At a reference noise level t_{ref}, we sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and form $\boldsymbol{x}_{t_{\mathrm{ref}}}^{\mathrm{ref}} = \alpha_{t_{\mathrm{ref}}} \boldsymbol{x}^{\mathrm{src}} + \sigma_{t_{\mathrm{ref}}} \boldsymbol{\epsilon}$. We then denoise this latent from t_{ref} to 0 under the target condition $\boldsymbol{c}_{\mathrm{tgt}}$. This auxiliary pass uses a short schedule independent of the editing trajectory, and its output is used only to construct the reference cache $\mathcal{C}$.

B Experimental Details

B.1 Implementation Details

Backbone and sampling configuration. We build Pace3D on the frozen `TRELLIS-image-large` checkpoint (Xiang et al., 2025) and the training-free flow editing (Ye et al., 2025), without optimizing any model parameters. Both stages use Euler integration with $\sigma_{\min} = 10^{-5}$, a timestep rescaling factor of 3, and classifier-free guidance for $t \in [0.5, 1]$. For SS editing, we use a 30-step schedule and start editing from step 10, with source and target guidance scales of 1 and 10, respectively. Unless otherwise stated, these settings are fixed across all experiments.

Proxy construction and phase control. We use a 30-step SS schedule and begin editing after skipping the first 10 steps. To construct the target proxy, we noise the source SS latent to this editing start point and denoise it for the remaining 20 steps under the target condition. We then interpolate between the source latent and the target proxy on the same sampling grid to obtain reference states for phase control. During editing, the current state is matched to the closest reference state, and the sampling phase is corrected when the mismatch exceeds 0.10. We allow at most $R_{\max} = 6$ phase corrections and apply Source Alignment with $\eta = 0.02$ throughout the editing process.

SLat generation and feature preservation. The SLat stage follows the pretrained 25-step sampling schedule with guidance scale 5. At each step, we additionally evaluate the source condition at the same noisy latent to estimate source–target feature discrepancy. We set $t_{\text{hi}} = 0.66$ and define token-wise edit evidence as the timestep-length-weighted mean relative response (Eq. (40)) over $t > t_{\text{hi}}$, with $\varepsilon = 10^{-6}$. The resulting evidence is smoothed on the 64^3 grid using a Gaussian kernel with standard deviation 7. By default, we set the release threshold to $\tau = 0.09$ and the transition half-width to $h = 0.08$ for constructing the preservation weights.

B.2 Evaluation Protocol

Benchmark and preprocessing. We evaluate on Eval3DEdit (Zhou et al., 2026), which covers diverse geometry and appearance edits. Following its protocol, each source asset is reconstructed from standardized multi-view renderings and encoded with the frozen `TRELLIS-image-large` model (Xiang et al., 2025). The same source assets, editing conditions, and preprocessing are used for all methods.

Geometry and target alignment. Following prior training-free 3D editing evaluations (Ye et al., 2025; Zhou et al., 2026), we represent each output by a normalized surface point cloud and evaluate it with Uni3D embeddings (Zhou et al., 2024). PC_{PC} measures geometric similarity to the source point cloud, while PC_{Img} and PC_{Txt} measure alignment with the target image and text instruction, respectively.

Rendered-view evaluation. Following the Eval3DEdit evaluation protocol (Zhou et al., 2026), we render each edited asset from a fixed set of turntable cameras under shared lighting and background settings. MV_{Img} and MV_{Txt} use CLIP features (Radford et al., 2021) to measure multi-view agreement with the target image and text. We additionally use DINOv2 features (Oquab et al., 2024) to compare the rendered output with the source and target reference images, reported as DINO_{src} and DINO_{tgt}. FID (Heusel et al., 2017) measures the distributional discrepancy between output and source renderings and is therefore reported as a source-fidelity metric.

Comparison protocol. Unless otherwise noted, baselines use their official implementations and default settings. All methods are evaluated from the same inputs using the same point-cloud normalization, cameras, renderer, and metric implementations. For each stochastic method, we report the mean over three random seeds.

C Structural-Lag Diagnostic Details

Data and case selection. We curate 12 editing cases from successful examples collected across several existing 3D editing methods. The cases are selected to contain clear and substantial structural changes, so that the evolution of decoded geometry and structural progress can be reliably analyzed. We additionally favor cases with spatially coherent edits and well-aligned source–target geometry, while excluding examples dominated by minor, fragmented, or ambiguous changes. These 12 cases are used specifically for the structural-progress analysis.

Trajectory and oracle reference. For the structural-lag analysis in Fig. 3, we trace the standard fixed-clock SS editing trajectory using 25 sampling steps, with editing starting from step 12. Each update averages five flow estimates. This yields 14 decoded states along each editing trajectory.

Under the same sampling schedule, we construct an oracle trajectory by interpolating between the paired source and ground-truth SS latents:

$$P_{t_k} = s_k \boldsymbol{x}^{\mathrm{src}} + (1 - s_k)\boldsymbol{x}^{\mathrm{gt}}, \qquad s_k = \frac{t_k}{t_{\mathrm{start}}}, \tag{45}$$

and decode it on the same time grid. Each state on the actual editing trajectory is matched to its nearest state on the oracle trajectory, and the difference between the corresponding phases defines the structural lag. The oracle trajectory requires the ground-truth latent and is used only for this diagnostic analysis. Figure 3 reports how this lag evolves during sampling and how the terminal lag relates to missing target structure.

Structural phase matching. Using the occupancy definition in Eq. (2), we measure the structural discrepancy between two decoded states by their Hamming distance:

$$d_{\mathrm{H}}(O, O') = \sum_j \mathbf{1}[O_j \neq O'_j] = \|O \oplus O'\|_1 . \tag{46}$$

For each editing state $\boldsymbol{x}_{t_i}$, we identify the closest phase on the oracle trajectory and define the structural lag as

$$\widehat{k}_i = \arg\min_k d_{\mathrm{H}}(O(\boldsymbol{x}_{t_i}), O(P_{t_k})) , \qquad \Delta t_i = t_{\widehat{k}_i} - t_i. \tag{47}$$

A positive Δt_i indicates that the current edit is structurally closer to an earlier oracle phase than implied by its sampling clock, i.e., structural progress is lagging behind elapsed sampling time. Figure 3(a) reports the median and interquartile range of this lag across cases.

Missing structure. Let $A = \Omega_{\mathrm{gt}} \backslash \Omega_{\mathrm{src}}$ and $D = \Omega_{\mathrm{src}} \backslash \Omega_{\mathrm{gt}}$ denote required additions and deletions. For the final fixed-clock support Ω_{FC}, the unresolved structural fraction is

$$M_{\mathrm{struct}} = \frac{|A \setminus \Omega_{\mathrm{FC}}| + |D \cap \Omega_{\mathrm{FC}}|}{|A| + |D|}. \tag{48}$$

Across the 12 cases, terminal lag is positive in every case, with a median of 0.486 and an interquartile range of $[0.345, 0.646]$. The unresolved structural fraction has a median of 0.702. Terminal lag is positively associated with missing structure, with a bootstrap 95% CI of $[0.83, 1.00]$, as shown in Fig. 3(b).

D Controller Analysis and Robustness

D.1 Does Proxy-Calibrated Phase Provide a Reliable Structural Progress Signal?

Table 5 compares Pace3D's decoder-space phase matching with the full-latent projection in Eq. (26) under the same sampling settings. As analyzed in Appendix A.2, full-latent projection is sensitive to off-path variation and to the scale of the source-to-proxy displacement, making the projected coordinate an unreliable measure of structural progress.

This limitation is reflected in the resulting editing behavior. Latent projection consumes a substantially larger SS budget, yet the additional sampling does not translate into consistent improvements

Table 5: **Latent-projection rollback diagnostic on Eval3DEdit.** Comparison of proxy-calibrated phase matching and full-latent projection under matched sampling settings. SS NFE excludes the common proxy pass.

		Source fidelity			Target alignment				
Estimator	**SS NFE** ↓	$\mathrm{PC}_{\mathrm{PC}}$ ↑	$\mathrm{DINO}_{\mathrm{src}}$ ↑	FID↓	$\mathrm{PC}_{\mathrm{Img}}$ ↑	$\mathrm{PC}_{\mathrm{Txt}}$ ↑	$\mathrm{MV}_{\mathrm{Img}}$ ↑	$\mathrm{MV}_{\mathrm{Txt}}$ ↑	$\mathrm{DINO}_{\mathrm{tgt}}$ ↑
Proxy-calibrated phase	**164.1**	**0.7997**	**0.8009**	**50.06**	**0.3653**	**0.2566**	**0.7291**	0.4899	**0.5733**
Latent projection	198.5	0.7817	0.7946	52.47	0.3604	0.2549	0.7227	**0.4907**	0.5675

in target alignment. Instead, source fidelity degrades across all three metrics. These results suggest that simply extending the trajectory based on latent-space projection does not provide an effective allocation of computation. Decoder-space matching more directly tracks changes in decoded structure, and we therefore use it for phase control in Pace3D.

D.2 Are the Gains Explained by Additional Inference Compute?

In this subsection, we examine whether Pace3D's improvements can be attributed simply to a larger inference budget rather than to its adaptive allocation. For both groups of baselines, we match Pace3D's average compute by adjusting only the NFE in the SS stage. Although Pace3D also includes a proxy-construction step, we exclude it from the compute count and measure only the NFE required to evolve from the source shape to the edited result, as this is the stage in which the actual editing is performed. Runtime, in contrast, is measured end-to-end from the algorithm input and therefore accounts for the SLat stage and all other model computations.

Table 6: **Controlling for inference compute on Eval3DEdit.** SS/shape NFE counts network forward evaluations along the editing trajectory and excludes Pace3D's proxy-construction pass. Runtime is averaged over edits. Best quality results within each panel are shown in **bold**.

				Source fidelity		Target alignment				
Method	**Budget setting**	**SS/shape NFE**	**Runtime (s)** ↓	$\mathrm{PC}_{\mathrm{PC}}$ ↑	$\mathrm{DINO}_{\mathrm{src}}$ ↑	$\mathrm{PC}_{\mathrm{Img}}$ ↑	$\mathrm{PC}_{\mathrm{Txt}}$ ↑	$\mathrm{MV}_{\mathrm{Img}}$ ↑	$\mathrm{MV}_{\mathrm{Txt}}$ ↑	$\mathrm{DINO}_{\mathrm{tgt}}$ ↑
(a) Cross-backbone compute controls										
PartFlow	default	100	16.97	**0.8154**	0.8058	0.3619	0.2473	0.7128	0.4776	0.5620
PartFlow	NFE matched	164	21.19	0.8131	**0.8073**	0.3606	0.2469	0.7135	0.4790	0.5637
AnchorFlow	default	160	21.28	0.6612	0.7489	0.3295	0.2324	0.6847	0.4712	0.5397
AnchorFlow	NFE matched	164	21.93	0.6336	0.7476	0.3286	0.2306	0.6759	0.4710	0.5314
Pace3D	adaptive	164.1	11.14	0.7997	0.8009	**0.3653**	**0.2566**	**0.7291**	**0.4899**	**0.5733**
(b) Same-backbone compute controls										
Nano3D	default	200	9.53	**0.8185**	0.7969	0.3545	0.2431	0.7083	0.4757	0.5604
Nano3D	NFE matched	164	8.50	0.7653	0.7918	0.3559	0.2468	0.7190	0.4844	0.5698
Nano3D	equal wall-clock	248	11.31	0.7736	0.7934	0.3574	0.2495	0.7202	0.4864	0.5711
Nano3D	5× sampling	820	31.3	0.7640	0.7915	0.3547	0.2477	0.7171	0.4848	0.5672
Pace3D	adaptive	164.1	11.14	0.7997	**0.8009**	**0.3653**	**0.2566**	**0.7291**	**0.4899**	**0.5733**

Table 6(a) compares Pace3D with cross-backbone controls under matched SS/shape NFE. Pace3D consistently achieves better target alignment than PartFlow and AnchorFlow. Although PartFlow attains higher source-fidelity scores, these metrics directly measure similarity to the source and thus also favor more conservative edits. Increasing the compute of PartFlow and AnchorFlow does not close the gap to Pace3D, indicating that the gains cannot be explained by additional inference compute alone.

Table 6(b) provides same-backbone controls with Nano3D. We first match the SS editing NFE by setting $n_{\mathrm{avg}} = 1$, consistent with Pace3D. Since Pace3D additionally incurs overhead from proxy construction and SLat score-difference computation, we also include an equal-wall-clock Nano3D variant with a comparable runtime. Pace3D consistently achieves better target alignment under both controls while maintaining stronger source fidelity. Moreover, increasing Nano3D to $5\times$ sampling ($n_{\mathrm{avg}} = 5$) still does not recover Pace3D's target-alignment performance. Together, these results suggest that the gains arise from adaptively allocating the editing horizon rather than simply using more inference compute.

Table 7: **Using the structural proxy as the SS output.**

	Source fidelity			Target alignment				
SS output	$\mathrm{PC}_{\mathrm{PC}}\uparrow$	$\mathrm{DINO}_{\mathrm{src}}\uparrow$	FID$\downarrow$	$\mathrm{PC}_{\mathrm{Img}}\uparrow$	$\mathrm{PC}_{\mathrm{Txt}}\uparrow$	$\mathrm{MV}_{\mathrm{Img}}\uparrow$	$\mathrm{MV}_{\mathrm{Txt}}\uparrow$	$\mathrm{DINO}_{\mathrm{tgt}}\uparrow$
Pace3D	0.7997	0.8009	50.06	0.3653	0.2566	0.7291	0.4899	0.5733
Proxy	0.7607	0.7911	54.21	0.3579	0.2497	0.7201	0.4815	0.5639

Table 8: **Proxy versus oracle control on the 12 paired cases.** FID is computed over 12 assets and is not comparable with Table 2. SS NFE excludes the proxy-construction pass.

		Source fidelity			Target alignment				
Ruler	**SS NFE** $\downarrow$	$\mathrm{PC}_{\mathrm{PC}}\uparrow$	$\mathrm{DINO}_{\mathrm{src}}\uparrow$	FID$\downarrow$	$\mathrm{PC}_{\mathrm{Img}}\uparrow$	$\mathrm{PC}_{\mathrm{Txt}}\uparrow$	$\mathrm{MV}_{\mathrm{Img}}\uparrow$	$\mathrm{MV}_{\mathrm{Txt}}\uparrow$	$\mathrm{DINO}_{\mathrm{tgt}}\uparrow$
Proxy (Pace3D)	179.9	0.6905	0.7550	120.09	**0.3957**	**0.2841**	0.7421	0.5012	0.7539
Oracle (ground truth)	**142.3**	**0.6964**	**0.7585**	**116.07**	0.3919	0.2792	**0.7432**	**0.5020**	**0.7543**

D.3 Why Use the Proxy as a Structural Ruler Rather Than an Editing Target?

We test whether the auxiliary SDEdit proxy can directly serve as the edited SS endpoint. Specifically, we feed $\boldsymbol{x}^{\mathrm{proxy}}$ into the same SLat stages, with all other settings unchanged. Table 7 reports three-seed averages on Eval3DEdit.

Using the proxy directly as the SS output consistently reduces source fidelity, while providing no measurable improvement in target alignment. This indicates that the proxy introduces additional structural drift without better realizing the requested edit.

This behavior is expected because the proxy is obtained by target-conditioned denoising from a substantially noised source latent, which can alter regions that should remain unchanged. Pace3D therefore uses the proxy only to estimate structural progress, rather than as the final editing result. The edited SS state remains on the source-coupled trajectory and is further stabilized by Source Alignment. In this role, the proxy only needs to provide a reliable ordering of structural progress, not a source-faithful endpoint.

D.4 Is the Proxy Accurate Enough for Horizon Control?

The proxy ruler only approximates true structural progress. To measure how much this matters, we use the 12 cases with paired ground-truth SS latents (Appendix C) and build an oracle ruler as in Eq. (7), with $\boldsymbol{x}^{\mathrm{proxy}}$ replaced by the ground-truth edited latent $\boldsymbol{x}^{\mathrm{gt}}$. Both rulers are matched to the same state with Eq. (9). The oracle controller uses this ruler for clock resets and keeps all other settings unchanged.

Oracle control yields comparable edits. Table 8 shows that the two controllers achieve comparable edit quality. The oracle provides slightly better source fidelity, whereas the proxy achieves better point-cloud target alignment, with no statistically significant difference favoring the oracle. The main advantage of the oracle is computational efficiency. It triggers fewer resets and shorter rewinds, resulting in a shorter overall SS trajectory.

The proxy is imperfect but sufficient for horizon control. At every SS update of the deployed trajectories, both rulers are matched to the same state, giving $\widehat{t}_{\mathrm{proxy}}$, $\widehat{t}_{\mathrm{oracle}}$, and the lags $\delta = \widehat{t} - t$ (Table 9). The phase error is $|\widehat{t}_{\mathrm{proxy}} - \widehat{t}_{\mathrm{oracle}}|$ averaged over the updates of each case, and the bias is the mean of $\widehat{t}_{\mathrm{proxy}} - \widehat{t}_{\mathrm{oracle}}$, which is positive when the proxy reports more lag. Spearman ρ is the rank correlation between δ_{proxy} and δ_{oracle} within each case, averaged over cases. Taking the oracle decision $\delta_{\mathrm{oracle}} > \theta$ as ground truth, AUC measures how well δ_{proxy} ranks the updates that the oracle would reset above the others, independent of the threshold, and F1 scores the proxy decision $\delta_{\mathrm{proxy}} > \theta$ at the controller's threshold. The positive bias shows that the proxy systematically reports more lag than the oracle, so its reset decisions agree with the oracle only in part and it triggers extra resets. The high AUC and positive rank correlation show that it still orders structural

Table 9: **Proxy phase estimates against the oracle ruler.**

Phase error↓			**Progress ordering↑**		**Decision↑**
Mean	Median	Bias	Spearman ρ	AUC	F1
0.133 [0.099, 0.166]	0.143 [0.093, 0.165]	+0.114 [0.065, 0.158]	0.628 [0.496, 0.746]	0.857 [0.799, 0.943]	0.592 [0.472, 0.696]

progress consistently with the oracle. Together with Table 8, the imprecision of the proxy mainly costs additional computation rather than editing quality.

E Parameter Analysis

Figure 5 analyzes the sensitivity of Pace3D to the maximum reset budget $R_{\max}$, the Source Alignment strength η, and the target-release threshold τ together with the transition half-width h of the preservation weights.

Reset budget. Increasing $R_{\max}$ from zero to a moderate budget consistently improves target alignment, after which the gains largely saturate. In contrast, source fidelity peaks at an intermediate budget and gradually deteriorates as more resets are allowed. This suggests that a small number of resets is sufficient to recover most of the target-alignment benefit, while overly frequent resets mainly introduce additional source deviation with limited semantic gain.

Source Alignment. The Source Alignment strength η exhibits a clear preservation–editing tradeoff. Increasing η initially improves source fidelity, while target alignment remains stable or slightly improves. Once the alignment constraint becomes too strong, however, target-side metrics degrade sharply, indicating that excessive source preservation can suppress the intended edit. Overall, moderate alignment provides effective source regularization without substantially restricting target realization.

CCC threshold and transition width. The release threshold τ and the transition half-width h jointly control how much target content is released at shared coordinates. Raising τ releases fewer tokens, leading to a lower FID and slight decreases in $\mathrm{MV}_{\mathrm{Img}}$, $\mathrm{MV}_{\mathrm{Txt}}$, and $\mathrm{DINO}_{\mathrm{tgt}}$. Within the tested range, h has a much smaller effect than τ, and the point-cloud metrics remain largely stable across all combinations.

F Additional Qualitative Results

Figure 6 covers diverse edits, including articulated motion, object addition, removal, replacement, and restyling. Pace3D realizes the requested structural and appearance changes—such as waving gestures, added or removed objects, replaced components, and style transformations—while retaining source identity and unaffected geometry. In comparison, competing methods more often leave the edit incomplete, alter surrounding structure, or introduce severe geometric artifacts. These results illustrate a consistent balance between edit realization and source preservation across edit types.

G Additional Quantitative Results

Table 10 reports paired 95% confidence intervals for the performance differences between Pace3D and each baseline in Table 2. We estimate intervals for case-level metrics by paired bootstrap over matched evaluation cases.

The intervals show broad and consistent improvements over weaker baselines. Comparisons with stronger hierarchical methods are closer, with target-alignment differences generally favoring Pace3D, while source-fidelity intervals often include zero. Overall, the results support improved target alignment without evidence of a systematic loss in source fidelity.

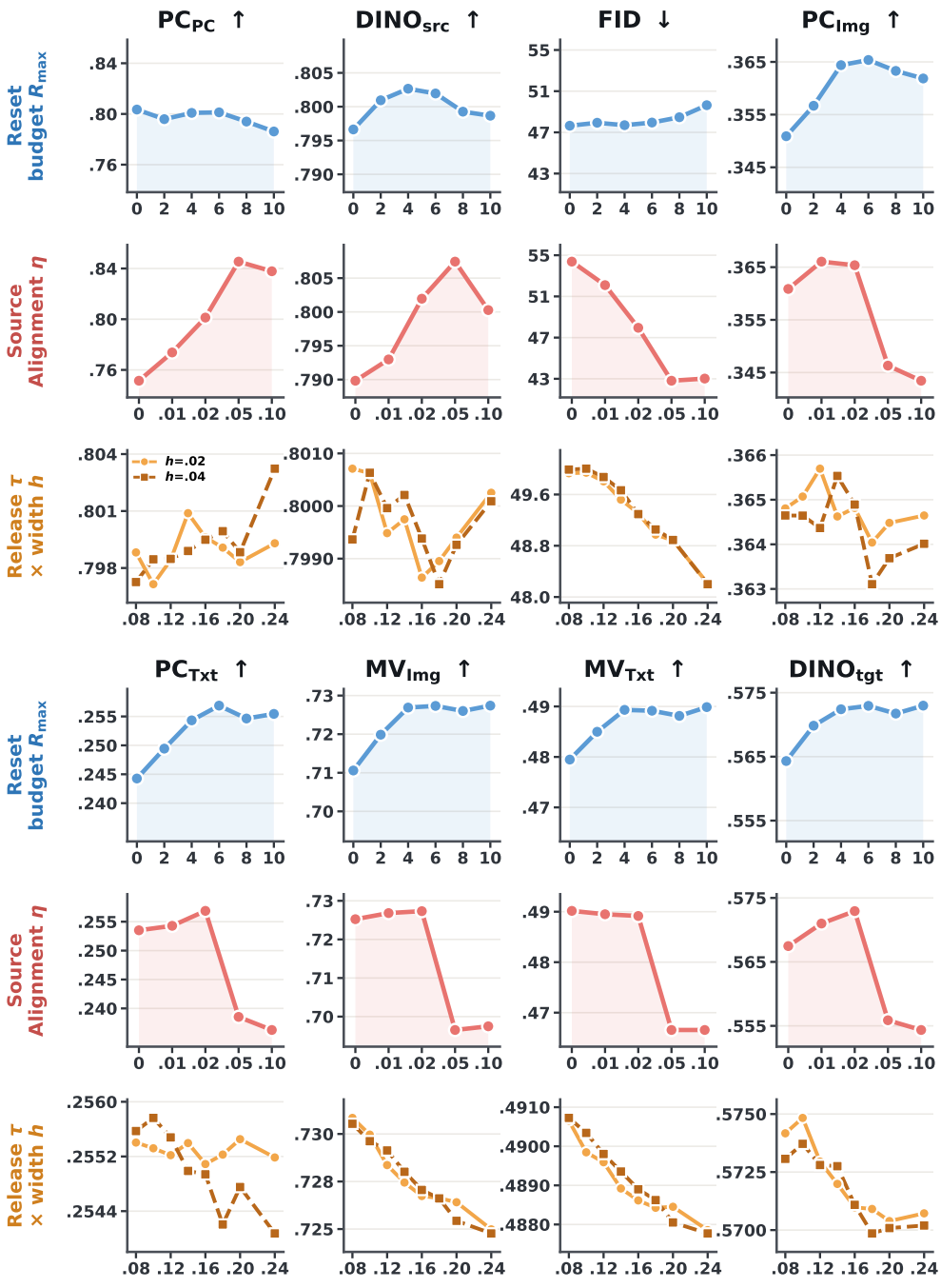


Figure 5: **Parameter sensitivity on Eval3DEdit.** We study the sensitivity to $R_{\max}$ and η by varying each parameter individually while keeping the others fixed, and jointly evaluate the release threshold τ and transition half-width h over $\tau \in [0.08, 0.24]$ and $h \in \{0.02, 0.04\}$. Results are reported across all eight evaluation metrics.

H Qualitative Ablation Results

Figure 7 complements the quantitative ablation in Table 3. For the pose edit, the baseline remains close to a T-pose and distorts the legs; the Controller realizes the requested pose but leaves a head-shape artifact, while Source Alignment corrects the face shape but changes its texture. For surfboard

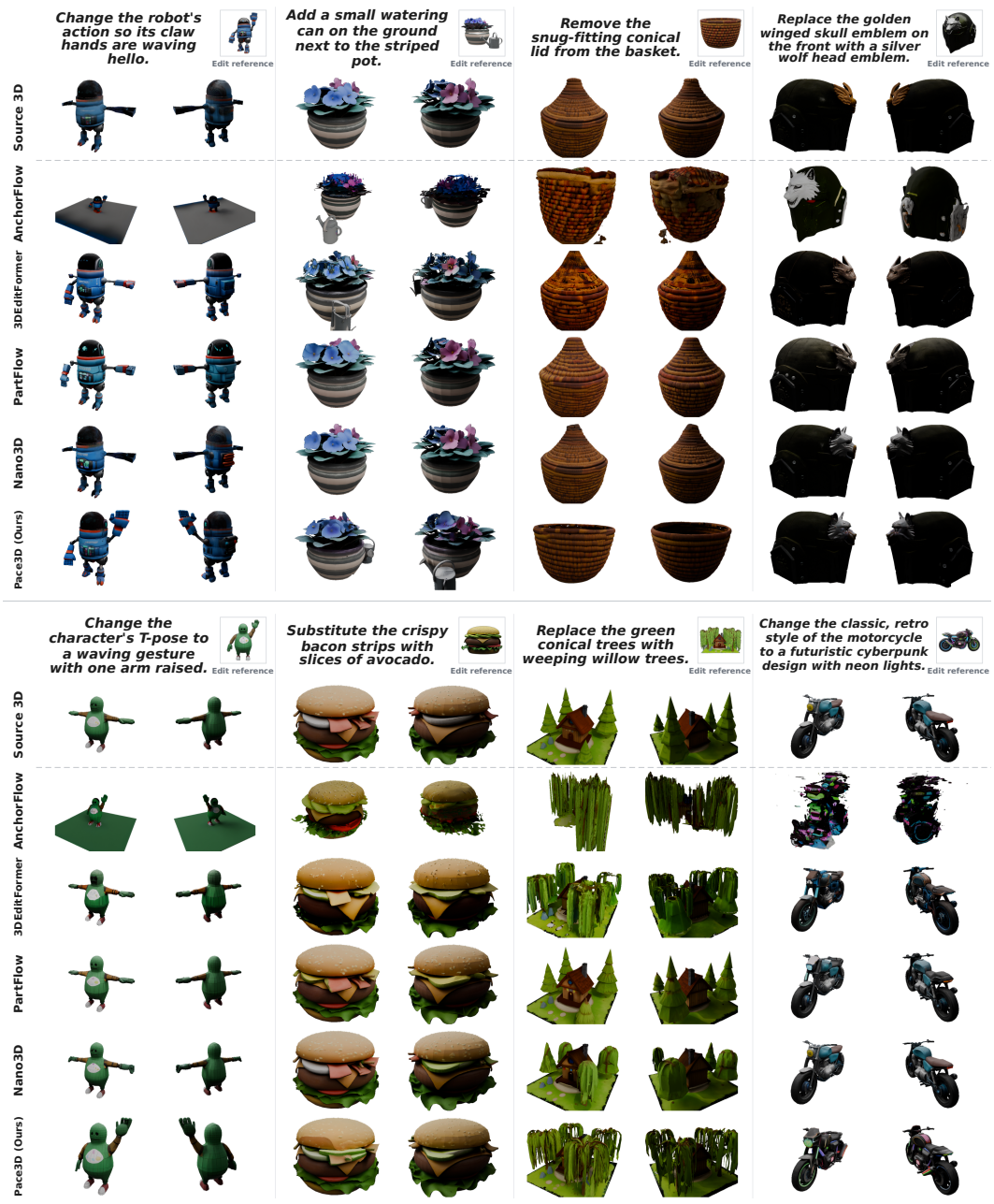


Figure 6: **Additional qualitative comparisons on Eval3DEdit.** We compare Pace3D with AnchorFlow, 3DEditFormer, PartFlow, and Nano3D across diverse editing cases, covering articulated motion, object addition, removal, replacement, and appearance restyling.

removal, the intermediate variants either leave the edit incomplete, introduce structural clutter, or alter the front texture. The full model completes both edits while preserving the relevant source structure and appearance.

I Details of User Study

We conduct a user study to assess the perceptual quality of the editing results. The study compares Pace3D with 3DEditFormer (Xia et al., 2025) and Nano3D (Ye et al., 2025) on 10 cases. 32 partici-

Table 10: **Paired 95% confidence intervals on Eval3DEdit.** Each cell shows the mean paired difference $\Delta = \text{Pace3D} - \text{baseline}$ (top) and its 95% bootstrap confidence interval (bottom). Positive values favor Pace3D. FID is omitted because it is a set-level statistic without per-case differences. Nano3D† is the SS-NFE-matched control from Table 2.

	Source fidelity		Target alignment				
Baseline	PC_{PC}	DINO_{src}	PC_{Img}	PC_{Txt}	MV_{Img}	MV_{Txt}	DINO_{tgt}
Edit360	+0.5290 [0.4916, 0.5659]	+0.1784 [0.1590, 0.1982]	+0.1775 [0.1619, 0.1928]	+0.1279 [0.1140, 0.1416]	+0.2001 [0.1862, 0.2137]	+0.0934 [0.0816, 0.1053]	+0.1507 [0.1360, 0.1658]
EditP23	+0.2499 [0.2078, 0.2928]	+0.0643 [0.0478, 0.0819]	+0.0783 [0.0656, 0.0915]	+0.0592 [0.0462, 0.0723]	+0.1015 [0.0890, 0.1143]	+0.0390 [0.0299, 0.0485]	+0.0811 [0.0705, 0.0926]
AnchorFlow	+0.1385 [0.1018, 0.1772]	+0.0521 [0.0343, 0.0727]	+0.0358 [0.0209, 0.0524]	+0.0242 [0.0126, 0.0363]	+0.0444 [0.0294, 0.0602]	+0.0186 [0.0099, 0.0277]	+0.0336 [0.0214, 0.0475]
3DEditFormer	+0.0009 [−0.0209, 0.0243]	+0.0059 [−0.0019, 0.0142]	+0.0075 [−0.0017, 0.0180]	+0.0101 [0.0032, 0.0176]	+0.0193 [0.0114, 0.0281]	+0.0128 [0.0072, 0.0188]	+0.0113 [0.0063, 0.0164]
PartFlow	−0.0228 [−0.0507, 0.0081]	−0.0066 [−0.0169, 0.0053]	+0.0019 [−0.0060, 0.0113]	+0.0093 [0.0016, 0.0181]	+0.0115 [0.0027, 0.0218]	+0.0122 [0.0058, 0.0191]	+0.0072 [0.0001, 0.0159]
Nano3D	−0.0188 [−0.0513, 0.0169]	+0.0040 [−0.0092, 0.0195]	+0.0108 [0.0005, 0.0228]	+0.0135 [0.0052, 0.0229]	+0.0209 [0.0081, 0.0356]	+0.0142 [0.0076, 0.0218]	+0.0129 [0.0026, 0.0251]
Nano3D†	+0.0344 [0.0127, 0.0616]	+0.0092 [−0.0008, 0.0221]	+0.0094 [0.0014, 0.0191]	+0.0098 [0.0011, 0.0203]	+0.0101 [0.0014, 0.0212]	+0.0055 [0.0002, 0.0121]	+0.0035 [−0.0036, 0.0128]

pants complete the study. For each case, the interface presents the editing prompt, two views of the source asset, and two corresponding views of each anonymized candidate. Method names and the edit reference image are not shown, and the case order is randomized for each participant.

For every case, participants select one preferred candidate under three criteria: *prompt alignment*, *visual quality*, and *content preservation*. Prompt alignment measures how faithfully the result performs the requested edit. Visual quality evaluates perceptual coherence and visible artifacts. Content preservation measures how well the result retains the source object's identity, semantic content, and untargeted attributes.

The preference rate in Table 1 is the fraction of these selections assigned to each method. Figure 8 shows the per-case preference distributions under all three criteria.

J Failure Cases and Limitations

Figure 9 illustrates two distinct sources of failure. Figure 9(a) shows reconstruction errors introduced by the SLat VAE itself. Even without editing, deterministic reconstruction can lose fine geometry and texture details, indicating a representation-level limitation that is inherited by downstream editing methods.

Figure 9(b) shows a failure of the structural proxy. Using the proxy directly as the SS output leads to severe geometric artifacts. Pace3D does not fully eliminate the difficulty of this case, but it produces a substantially more coherent result because the proxy is used only as a reference for estimating structural progress, while the edited state remains on the source-coupled trajectory. This separation limits the extent to which proxy artifacts are transferred to the final output.

The first two failures occur before the controller is invoked and bound the source fidelity attainable by any downstream trajectory using this latent representation. The third shows that an unreliable proxy can make phase estimation less precise, but its geometric defects need not transfer directly to the output. This robustness comes from using the proxy only for structural guidance rather than directly modifying the edited state. It cannot, however, recover information lost by the VAE or compensate for a severely inaccurate proxy.

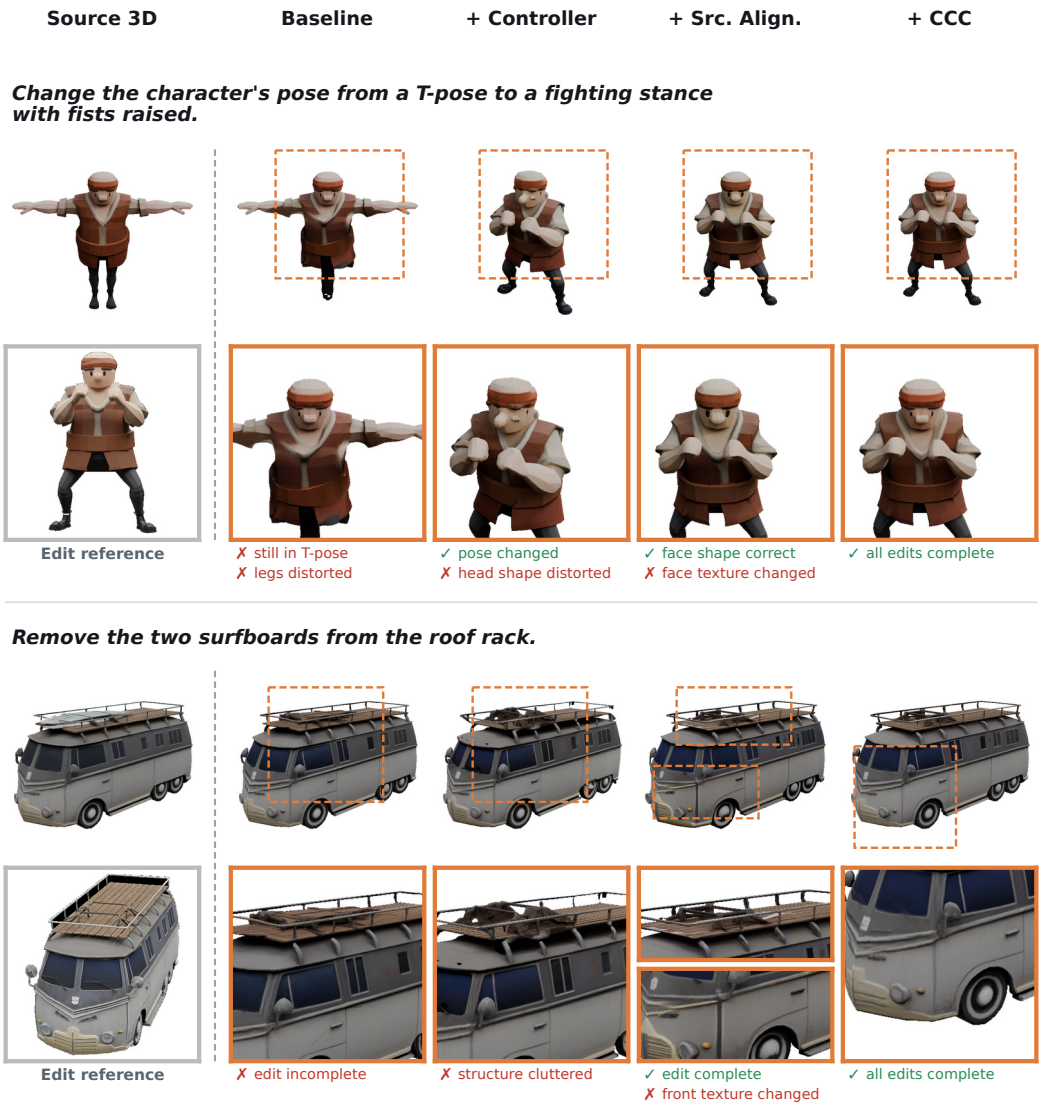


Figure 7: **Qualitative visualization of the component ablation.** Each row shows the source, the edit reference, and the outputs obtained as the Controller, Source Alignment, and Cross-Condition Consensus (CCC) are introduced.

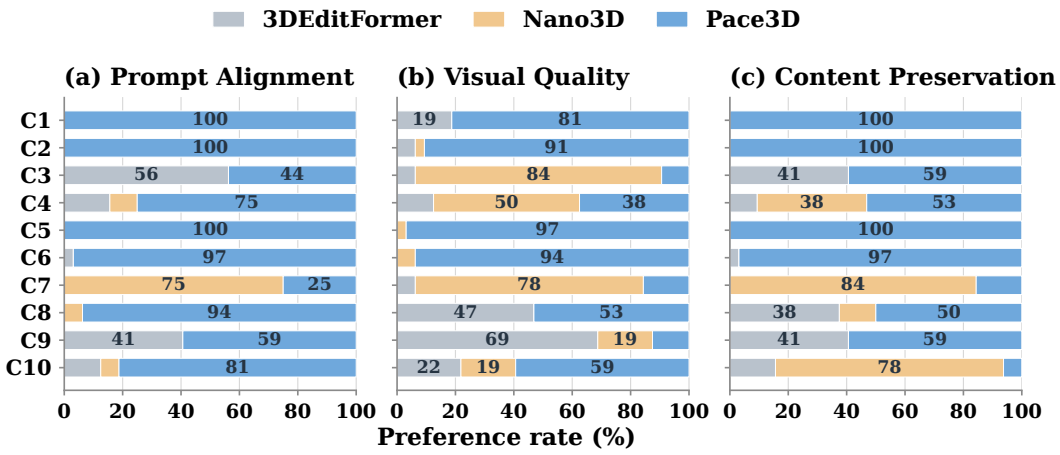


Figure 8: **User-study preference statistics.** Per-case preference distributions for (a) prompt alignment, (b) visual quality, and (c) content preservation. Each stacked bar aggregates 32 selections.

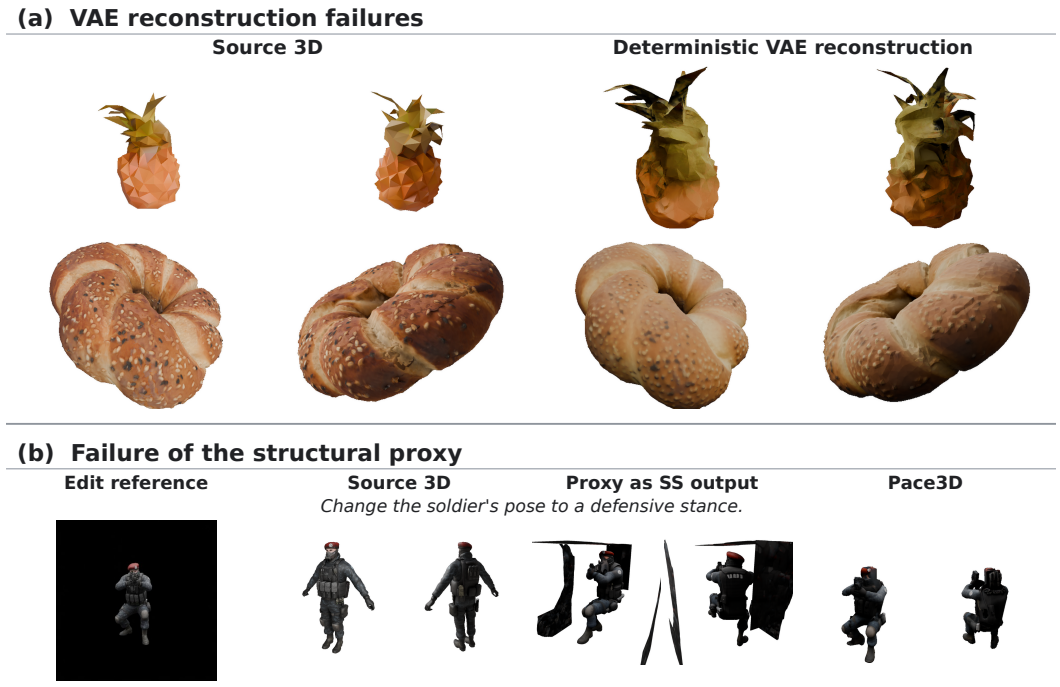


Figure 9: **Failure cases and their sources.** (a) Two deterministic source reconstructions through the SLat VAE, with posterior sampling disabled and no editing or controller, expose loss of fine geometry and texture. (b) When used directly as the SS output, a severely distorted proxy produces floating artifacts. Pace3D uses the same proxy only as a structural ruler and still obtains a coherent defensive pose.