CACHE3D: ADAPTIVE CACHING FOR FASTER STRUCTURED 3D LATENT DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion and rectified-flow models enable high-quality 3D synthesis, but their iterative inference remains computationally expensive. Transferring existing caching strategies to structured latent 3D generation poses reliability challenges: globally aggregated change estimates can obscure spatially concentrated latent evolution, while a straightforward token-output reuse adaptation mixes stale predictions with outputs recomputed under an altered attention context. We propose **Cache3D**, a training-free acceleration framework that separates the spatial granularity of cache-reliability estimation from the computational granularity of reuse. Cache3D groups sparse tokens into cells of the latent 3D grid, averages anchor-relative token drift within each cell, and uses the maximum cell response as a localized spatial risk metric. This signal guides adaptive selection among denoising-step reuse, transformer-block fallback, and full denoiser refresh, without treating individual tokens or cells as independent cache units. On the TRELLIS 2.0 example-image benchmark, Cache3D achieves **1.633**$\times$ end-to-end speedup and reduces shape latent relative RMSE by **16.0%** compared with EasyCache, while maintaining comparable geometric fidelity.

1 INTRODUCTION

Recent 3D diffusion and rectified-flow models have demonstrated strong capabilities in generating high-quality 3D assets from text and images (Xiang et al., 2024; 2025; Chen et al., 2024b; Zhao et al., 2025). By combining structured latent representations with transformer-based generative backbones, these models enable scalable and high-fidelity 3D synthesis. However, such advances come at substantial inference cost. Generating a single 3D asset typically requires dozens of iterative sampling steps, each involving the evaluation of a large transformer network. As a result, repeated backbone evaluation becomes a major computational bottleneck, motivating efficient acceleration strategies that reduce redundant computation while preserving generation fidelity.

Caching-based acceleration reduces this cost by exploiting redundancy between nearby sampling states in image and video diffusion models. These approaches typically operate at either the step-level, skipping entire model evaluations or blocks when trajectory changes are small (Liu et al., 2024; Zhou et al., 2025; Lyu et al., 2025; Huang et al., 2025; Ma et al., 2024); or the token-level, reusing fine-grained attention maps and token representations for static regions (Zou et al., 2025; Lou et al., 2024). While these methods have inspired initial explorations in 3D generation (Yang et al., 2026), directly transferring them to sparse structured 3D latents presents unique challenges. In sparse structured 3D latents, tokens are associated with discrete spatial coordinates on a latent 3D grid before being decoded into surfaces. Consequently, inaccuracies during cache reuse can suppress localized, high-frequency updates, causing geometric artifacts such as broken thin structures, missing components, or topological errors.

Specifically, step-level methods evaluate redundancy by pooling changes globally across the latent representation. This metric can be dominated by large stable regions, masking localized changes and leading to reuse decisions that fail to refresh denoising predictions for delicate structures that have not yet stabilized. Conversely, a direct adaptation based on token-output reuse introduces a consistency problem: selected tokens are recomputed using only the selected token subset, under an attention context different from that of the full model. The resulting composite output combines stale and fresh predictions from conflicting sampling states. Advancing the sampler with this mismatched

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.

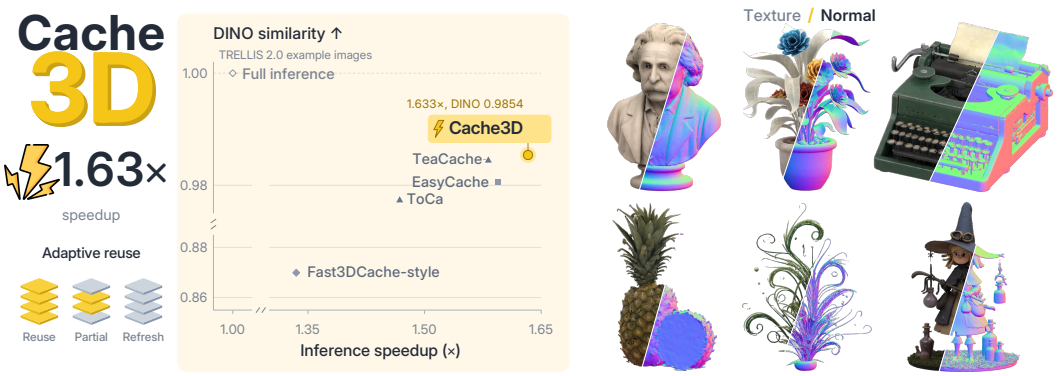

Figure 1: **Cache3D accelerates structured latent 3D generation.** Left: Cache3D achieves a favorable trade-off between speedup and visual fidelity compared to existing caching methods, reaching a $1.63\times$ speedup while maintaining high rendered-view similarity to full inference on the TRELLIS 2.0 example-image set. Right: Representative generated 3D assets shown with textured and surface-normal renderings, effectively preserving delicate surfaces, thin components, and intricate geometry without introducing topological artifacts.

representation introduces discrepancies that compound across subsequent steps, ultimately degrading surface geometry and causing topological artifacts. These complementary limitations indicate that reliable 3D caching must evaluate reuse risk with fine spatial sensitivity without resorting to piecemeal token stitching. Therefore, we are motivated to separate the granularity of reliability estimation from that of computation reuse.

To resolve this challenge, we propose **Cache3D**, a training-free acceleration framework for geometry-preserving, structured latent 3D diffusion. The pipeline is illustrated in Fig. 2. Cache3D decouples risk estimation from execution by evaluating reliability at a localized cell-level while retaining unified model-level updates. Specifically, it partitions sparse tokens into axis-aligned cells over the 3D coordinate grid and tracks individual token drift across timesteps. By averaging responses within each cell and taking the maximum across all cells, our metric remains sensitive to localized structural changes without being misled by global averages or destabilized by isolated token noise. Crucially, these cells serve solely as a spatial measurement grid rather than units for computation reuse, completely sidestepping context fragmentation. Guided by this spatially aware indicator, Cache3D dynamically selects among full-step reuse, block-level fallback, and full model refresh, achieving substantial acceleration while safe guarding geometric fidelity. As shown in Fig. 1, Cache3D achieves a favorable efficiency–geometry trade-off on structured 3D generation benchmarks, accelerating inference by $1.63\times$ while preserving high geometric quality and avoiding the severe artifacts caused by token-level reuse.

2 Related Work

3D Diffusion and Flow-based Generative Models. Early text-to-3D methods optimize individual assets using guidance from pretrained 2D diffusion models, including score distillation sampling and its variants (Poole et al., 2023; Lin et al., 2023; Wang et al., 2023; Chen et al., 2023). Although effective for open-vocabulary generation, these approaches require expensive per-instance optimization. With the growth of large-scale 3D datasets such as Objaverse and Objaverse-XL (Deitke et al., 2023b;a), recent work has shifted toward direct 3D generative models based on diffusion or rectified flow (Nichol et al., 2022; Liu et al., 2023; Xiang et al., 2024; Chen et al., 2024b; Zhao et al., 2025).

Within this direction, sparse structured latents provide an efficient representation for scalable 3D generation. By assigning tokens only to active spatial regions, they avoid processing the entire dense 3D grid and reduce the number of tokens required for transformer modeling. Meanwhile, their explicit grid coordinates preserve local spatial relationships between latent features. TRELLIS uses this design to encode geometry and appearance for text- and image-conditioned generation (Xiang et al., 2024), while TRELLIS 2.0 (Xiang et al., 2025) further develops a compact native structured

latent space. These properties make TRELLIS 2.0 a natural backbone for Cache3D, which estimates cache reliability from localized latent evolution.

Diffusion Acceleration. Diffusion inference has been accelerated through distillation, consistency models, faster samplers, pruning, and attention optimization (Song et al., 2023; Salimans & Ho, 2022; Luo et al., 2023; Lu et al., 2022; Chen et al., 2024a). Among methods requiring no additional training or optimization, caching exploits redundancy between adjacent denoising states. FORA reuses attention and MLP outputs across sampling steps (Selvaraju et al., 2024), while TeaCache and EasyCache estimate timestep-level redundancy using inexpensive runtime signals (Liu et al., 2024; Zhou et al., 2025). FasterCache combines dynamic feature reuse with classifier-free-guidance caching for video diffusion (Lyu et al., 2025), whereas ToCa performs selective token-wise feature caching (Zou et al., 2025). Other methods mitigate accumulated reuse error or optimize non-uniform caching schedules (Qiu et al., 2025; Cao et al., 2026). Learning-to-Cache, HarmoniCa, and TokenCache instead introduce router optimization, cache-aware training, or post-training predictors (Ma et al., 2024; Huang et al., 2025; Lou et al., 2024). These methods primarily focus on image and video diffusion rather than spatially structured 3D latents.

Caching for 3D Generation. Fast3DCache extends training-free caching to 3D geometry generation by using voxel stabilization patterns to schedule cache quotas and velocity-based spatiotemporal criteria to select stable voxel features for reuse (Yang et al., 2026). Cache3D shares the goal of accelerating 3D generation but separates the spatial granularity of reliability estimation from the granularity of computation reuse. Specifically, spatial cells are used to detect localized latent evolution, while cached computation is reused at the denoising-step or transformer-block level rather than through direct token or voxel replacement.

3 METHOD

Structured latent 3D generation requires repeatedly evaluating a heavy transformer backbone during flow sampling, incurring substantial compute cost. While feature caching offers an attractive training-free remedy, standard paradigms struggle with sparse 3D latents: whole-step reuse overlooks localized geometric evolution, whereas token-level caching disrupts self-attention consistency. To overcome this tension, we propose Cache3D, an acceleration framework that decouples fine-grained spatial risk detection from unified model-level execution. An overview of the proposed pipeline is shown in Fig. 2. We first formulate the sampling dynamics and define three runtime execution modes in Sec. 3.1. We then introduce a spatially pooled drift metric that governs mode allocation in Sec. 3.2, followed by our hierarchical residual reuse mechanism and cache update protocol in Sec. 3.3.

3.1 PROBLEM SETUP AND OVERVIEW

Modern structured 3D generation frameworks such as Xiang et al. (2025) synthesize assets using pretrained flow-matching transformers. Let $\{t_k\}_{k=0}^{K}$ with $t_0 > t_1 > \cdots > t_K$ denote the discretized sampling schedule, and let $\mathbf{x}_k = \mathbf{x}_{t_k}$ be the structured 3D latent at step k. Conditioned on input context $\mathbf{c}$, the transformer F_θ predicts a velocity field used by an Euler numerical sampler to advance the latent:

$$\widehat{\mathbf{v}}_k = F_\theta(\mathbf{x}_k, t_k; \mathbf{c}), \qquad \mathbf{x}_{k+1} = \mathbf{x}_k - \Delta t_k \widehat{\mathbf{v}}_k, \quad \text{where } \Delta t_k = t_k - t_{k+1}. \tag{1}$$

Standard inference evaluates F_θ across all K steps. We eliminate this redundant compute by adaptively routing each step through one of three execution modes:

- **Full Refresh:** Evaluates the transformer backbone F_θ and synchronizes all cache states.
- **Block Fallback:** Partially executes the network by recomputing volatile blocks while reusing cached transformations for stable blocks.
- **Step Reuse:** Completely bypasses the transformer, extrapolating velocity via cached residuals at zero neural compute cost.

Whenever a network forward pass is performed (under Full Refresh or Block Fallback), we record that step index $\tau \leq k$ and store the current latent $\mathbf{x}_\tau$ as a reference state anchor $\mathbf{A}_\tau$ (i.e. $\mathbf{A}_\tau = \mathbf{x}_\tau$).

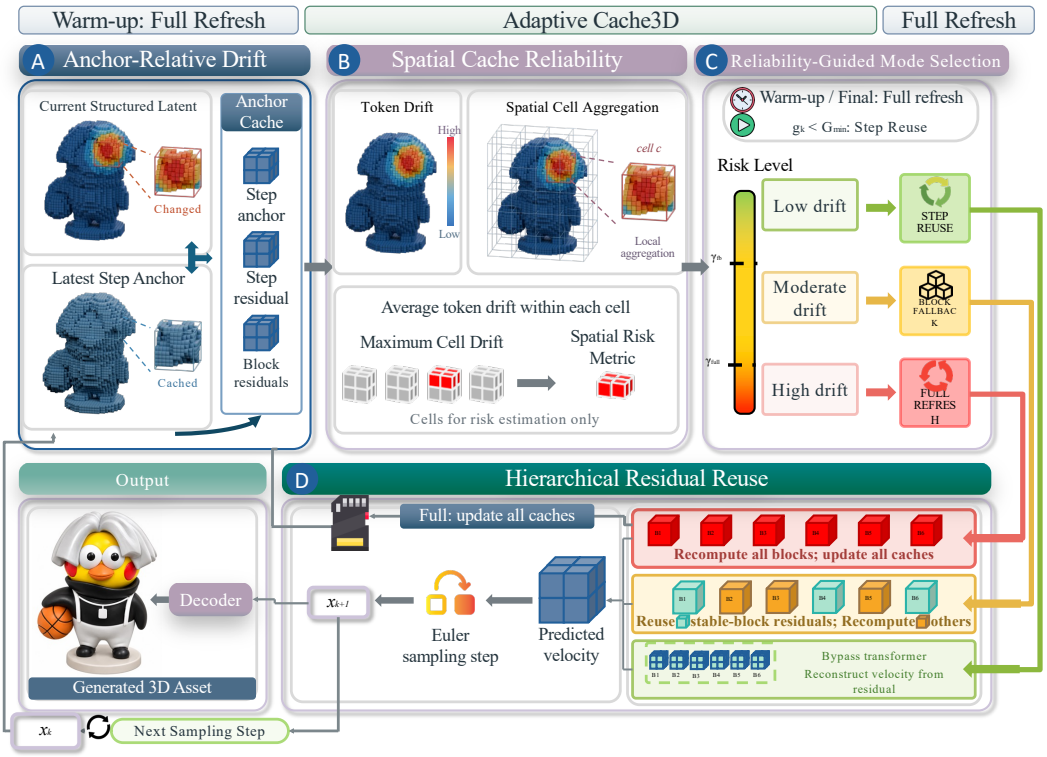

Figure 2: **Overview of Cache3D.** (A) Given a structured 3D latent, we compute token-level drift relative to the latest reference anchor and pool it into spatial cells. (B–C) The maximum cell drift indicates how far the latent trajectory has drifted and serves as a spatial risk metric to adaptively route sampling through one of three execution modes: FULL REFRESH, BLOCK FALLBACK, or STEP REUSE. (D) Under STEP REUSE, the transformer is bypassed entirely, extrapolating velocity via a cached velocity–latent residual at zero neural compute cost. Under BLOCK FALLBACK, volatile transformer blocks are recomputed while cached transformations for stable blocks are reused. Finally, states and residuals are synchronized following the cache update protocol to advance the sampler to the next step.

The validity of reusing cached computation at subsequent steps depends strictly on how far the current latent trajectory $\mathbf{x}_k$ has drifted from $\mathbf{A}_\tau$.

To govern this reuse reliably, we monitor this drift across localized *cells* of the latent 3D coordinate grid. Crucially, these cells serve purely as a spatial risk metric rather than independent cache partitions, decoupling fine-grained error detection from unified model-level execution.

3.2 SPATIALLY STRUCTURED CACHE RELIABILITY

To decide when cached states remain valid throughout sampling, we require a metric that tracks localized latent divergence without incurring full model evaluations. To this end, we aggregate token-level drift across spatial coordinates (Fig. 2 (A–B)), and map this signal to our execution modes (Fig. 2 (C)).

Spatial Drift Aggregation. Let $\mathbf{x}_k \in \mathbb{R}^{N\times d}$ denote the structured latent at step k, comprising N sparse latent tokens whose feature representations are denoted by $\{\mathbf{x}_{k,i}\}_{i=1}^{N} \subset \mathbb{R}^d$. Each token i is associated with a discrete spatial coordinate $\mathbf{p}_i = (x_i, y_i, z_i) \in \mathbb{Z}^3$ on the latent 3D grid. Across a sampling rollout, token coordinates and sequence indexing remain invariant; only the feature representations evolve over time. Let $\mathbf{A}_\tau = \mathbf{x}_\tau$ denote the anchor latent recorded at the most recent network execution step τ, such that $\mathbf{A}_{\tau,i} = \mathbf{x}_{\tau,i}$. Consequently, the feature vector $\mathbf{x}_{k,i}$ and its corresponding anchor vector $\mathbf{A}_{\tau,i}$ represent identical spatial locations in the 3D volume. We quantify

the relative feature drift of token i from its anchor state $\mathbf{A}_{\tau,i}$ at step k as:

$$\delta_i(k,\tau) = \frac{\mathrm{RMS}\left(\mathbf{x}_{k,i} - \mathbf{A}_{\tau,i}\right)}{\mathrm{RMS}\left(\mathbf{A}_{\tau,i}\right) + \epsilon}, \tag{2}$$

where $\mathrm{RMS}(\mathbf{z}) = \sqrt{d^{-1}\sum_{j=1}^{d} z_j^2}$, and $\epsilon > 0$ ensures numerical stability.

In structured 3D generation, evaluating δ_i via a global average dilutes localized high-frequency updates across large static background regions. Conversely, tracking an unconstrained single-token maximum exposes the schedule to high-frequency stochastic outliers. To capture coherent localized evolution, we partition the sparse token volume into axis-aligned spatial *cells* of width s. Token i is assigned to cell coordinate $\mathbf{q}_i$:

$$\mathbf{q}_i = \left\lfloor \frac{\mathbf{p}_i}{s} \right\rfloor, \qquad \mathcal{I}_c = \{i \mid \mathbf{q}_i = c\}, \tag{3}$$

where $\lfloor\cdot\rfloor$ operates element-wise and $\mathcal{I}_c$ indexes all active sparse tokens mapped to cell c. Because the underlying 3D representation is sparse, the token cardinality $|\mathcal{I}_c|$ varies across cells.

To suppress single-token noise while remaining sensitive to local updates, we pool token drift into cell-level risk in two stages:

$$D_c(k,\tau) = \frac{1}{|\mathcal{I}_c|}\sum_{i\in\mathcal{I}_c}\delta_i(k,\tau), \qquad D_{\mathrm{cell}}(k,\tau) = \max_{c\in\mathcal{C}} D_c(k,\tau), \tag{4}$$

where $\mathcal{C}$ denotes the set of occupied spatial cells. Here, $D_c(k,\tau)$ measures the average token drift within cell c, while $D_{\mathrm{cell}}(k,\tau)$ records the maximum cell drift across the whole volume. For cells containing multiple tokens, the local average reduces the impact of single noisy tokens and highlights updates shared across the local latent neighborhood, while the subsequent max-pooling across cells captures the strongest local change.

Reliability-Guided Mode Selection. We map the cell drift metric D_{cell} to the three execution modes introduced in Sec. 3.1 (Full Refresh, Block Fallback, and Step Reuse) using an adaptive thresholding schedule governed by two thresholds, $\gamma_{\mathrm{fb}} < \gamma_{\mathrm{full}}$. To stabilize initial trajectory formation and preserve fine details during terminal convergence, we execute a Full Refresh during predefined warm-up and final sampling steps.

For all intermediate steps, let $g_k = k - \tau$ denote the steps elapsed since the last network forward pass (where either Block Fallback or Full Refresh resets the anchor $\tau \leftarrow k$). Once a minimum interval $G_{\min} \geq 1$ is reached to avoid rapid mode transitions, the execution mode is determined by:

$$\mathrm{mode}(k) = \begin{cases} \mathrm{FullRefresh}, & g_k \geq G_{\min} \ \text{ and } \ D_{\mathrm{cell}}(k,\tau) \geq \gamma_{\mathrm{full}}, \\ \mathrm{BlockFallback}, & g_k \geq G_{\min} \ \text{ and } \ \gamma_{\mathrm{fb}} \leq D_{\mathrm{cell}}(k,\tau) < \gamma_{\mathrm{full}}, \\ \mathrm{StepReuse}, & \text{otherwise.} \end{cases} \tag{5}$$

3.3 Hierarchical Residual Reuse

Guided by the mode selection rule, we execute computation across multiple granularities to maximize throughput while preserving geometric fidelity. Depending on the evaluated risk, we avoid redundant forward passes through cached step-level trajectory updates or selective block-level re-evaluation (Fig. 2 (D)).

Step-Level Residual Reuse. When $\mathrm{StepReuse}$ is selected, the transformer backbone is bypassed entirely. Rather than reusing a stale velocity $\overline{\mathbf{v}}_\tau$, we update the velocity estimate using a cached *velocity–latent residual* $\mathbf{r}_\tau = \overline{\mathbf{v}}_\tau - \mathbf{A}_\tau$:

$$\widetilde{\mathbf{v}}_k = \mathbf{x}_k + \mathbf{r}_\tau = \overline{\mathbf{v}}_\tau + (\mathbf{x}_k - \mathbf{A}_\tau). \tag{6}$$

This formulation incorporates the trajectory displacement $(\mathbf{x}_k - \mathbf{A}_\tau)$ directly into the velocity update at zero network evaluation cost, after which the state advances via $\mathbf{x}_{k+1} = \mathbf{x}_k - \Delta t_k \widetilde{\mathbf{v}}_k$.

Block-Level Residual Fallback. When cell risk triggers Block Fallback, we recover computation by evaluating volatile transformer blocks while reusing cached transformations for stable blocks. Let $\mathbf{H}^{\text{in}}_{b,k}$ and $\mathbf{H}^{\text{out}}_{b,k}$ denote the input and output activations for block $b \in \mathcal{B}$ at step k. Rather than caching absolute output activations, we cache the intra-block transformation residual:

$$\mathbf{R}_{b,k} = \mathbf{H}^{\text{out}}_{b,k} - \mathbf{H}^{\text{in}}_{b,k}, \qquad \widetilde{\mathbf{H}}^{\text{out}}_{b,k} = \mathbf{H}^{\text{in}}_{b,k} + \mathbf{R}^{\text{cache}}_{b,\tau_b}, \tag{7}$$

where τ_b indexes the most recent Full Refresh step that evaluated block b.

To identify stable blocks, we track the temporal variance of each block's residual during full evaluations. The instantaneous relative variation $e_{b,k}$ and its exponential moving average (EMA) s_b are computed as:

$$e_{b,k} = \frac{\text{RMS}\left(\mathbf{R}^{\text{new}}_{b,k} - \mathbf{R}^{\text{cache}}_{b,\tau_b}\right)}{\max\left\{\text{RMS}\left(\mathbf{R}^{\text{new}}_{b,k}\right), \epsilon_b\right\}}, \qquad s_b \leftarrow \beta s_b + (1-\beta) e_{b,k}, \tag{8}$$

where $\beta \in [0, 1)$ is the momentum parameter and $\epsilon_b > 0$ avoids division by zero. Blocks whose smoothed variation remains within a multiplicative tolerance m of the minimum variation are reused:

$$\mathcal{B}_{\text{reuse}} = \left\{ b \in \mathcal{B} \;\middle|\; s_b \le m \cdot \max\left(\min_{j \in \mathcal{B}} s_j,\, \epsilon_s\right) \right\}, \tag{9}$$

where ϵ_s safeguards against degenerate zero-variance floors. Blocks in $\mathcal{B}_{\text{reuse}}$ apply Eq. 7, while active blocks $\mathcal{B} \setminus \mathcal{B}_{\text{reuse}}$ undergo standard forward evaluation. In short, D_{cell} governs *when* fallback occurs, while s_b determines *which* network depth to execute.

Cache Update Protocol. Cached states are updated differently across the three modes:

- **Full Refresh:** Synchronizes the step anchor $\mathbf{A}_\tau \leftarrow \mathbf{x}_k$, updates the velocity residual $\mathbf{r}_\tau \leftarrow \overline{\mathbf{v}}_k - \mathbf{A}_\tau$, and overwrites all block residuals $\mathbf{R}^{\text{cache}}_{b,\tau_b}$ alongside their EMA scores s_b.
- **Block Fallback:** Updates the step anchor $\mathbf{A}_\tau \leftarrow \mathbf{x}_k$ and residual $\mathbf{r}_\tau \leftarrow \overline{\mathbf{v}}_k - \mathbf{A}_\tau$ using the fallback velocity output, leaving block-level caches $\mathbf{R}^{\text{cache}}_{b,\tau_b}$ and scores s_b untouched.
- **Step Reuse:** Advances the sampler in read-only mode without altering cache records.

All cache buffers are purged between generation stages or whenever the underlying sparse token coordinate layout changes.

4 Experiments

In this section, we present experimental details and evaluations to demonstrate the effectiveness of Cache3D. Sec. 4.1 introduces the implementation and evaluation protocols, while Sec. 4.2 reports quantitative comparisons, qualitative results, ablation studies, and analyzes different cache granularities.

4.1 Implementation Details

Experimental Setup. We evaluate Cache3D on TRELLIS 2.0 (Xiang et al., 2025) using its official example images and Toys4K (Stojanov et al., 2021). All methods share pretrained weights, inputs, sampling schedules, random seeds, and decoding pipelines. Stage-specific caching hyperparameters are provided in the Appendix.

Baselines. We compare against full inference, EasyCache (Zhou et al., 2025), TeaCache (Liu et al., 2024), and token-output reuse adaptations inspired by ToCa (Zou et al., 2025) and Fast3DCache (Yang et al., 2026). The token-output adaptations recompute selected token subsets and reuse previous velocity predictions for the remaining tokens; they are not full reproductions of the original methods. Direct token and cell reuse are additionally evaluated as diagnostic variants.

Table 1: **Cache-induced degradation on the example-image benchmark.** Metrics are measured against full inference under the same input and seed. Speedup includes input conditioning and all generative sampling stages. Lower RMSE, CD, and LPIPS are better; higher F-score and DINO similarity are better. **Bold** and underlined values indicate the best and second-best results, respectively.

Method	Reuse	Speedup ↑	Rel. RMSE ↓	CD ↓	F@0.02 ↑	LPIPS ↓	DINO ↑
EasyCache	timestep	1.594×	0.3769	0.00934	0.9631	0.0681	0.9806
TeaCache	timestep	1.582×	0.3435	**0.00931**	0.9636	0.0597	0.9846
ToCa	module/token	1.468×	0.6688	0.00956	0.9520	0.0828	0.9775
Fast3DCache-style	token	1.335×	0.7900	0.01078	0.9095	0.1928	0.8699
Cache3D	step+block	**1.633**×	**0.3167**	**0.00931**	**0.9637**	**0.0561**	**0.9854**

Table 2: **Cache-induced degradation on Toys4K.** Metrics are measured against full inference under the same input and seed. Speedup includes input conditioning and all generative sampling stages. Lower RMSE, CD, and LPIPS are better; higher F-score and DINO similarity are better. **Bold** and underlined values indicate the best and second-best results, respectively.

Method	Reuse	Speedup ↑	Rel. RMSE ↓	CD ↓	F@0.02 ↑	LPIPS ↓	DINO ↑
EasyCache	timestep	1.732×	0.3291	0.01166	0.9010	0.0531	0.9743
TeaCache	timestep	**1.744**×	0.3121	0.01165	0.9016	**0.0495**	**0.9792**
ToCa	module/token	1.589×	0.6054	0.01319	0.8573	0.3869	0.8055
Fast3DCache-style	token	1.451×	0.7890	0.01410	0.8090	0.4274	0.7264
Cache3D	step+block	1.739×	**0.3077**	**0.01164**	**0.9017**	0.0525	0.9778

Evaluation Metrics. We report end-to-end runtime and speedup, together with fidelity to full inference measured by shape latent relative RMSE, mesh Chamfer Distance (CD) and F-score at a threshold of 0.02 (Fan et al., 2017; Tatarchenko et al., 2019), and rendered-view LPIPS (Zhang et al., 2018) and DINO similarity. Detailed metric and rendering protocols are provided in the Appendix.

4.2 Experimental Results

Quantitative Comparison. Tab. 1& 2 report the quantitative comparison on the example-image benchmark and Toys4K, respectively. On the example-image benchmark, Cache3D achieves a $1.633\times$ speedup while obtaining the lowest shape relative RMSE among the accelerated methods. Compared with EasyCache, Cache3D reduces the RMSE from 0.3769 to 0.3167, while maintaining comparable geometric quality in CD and F-score. This shows that Cache3D introduces less latent-space deviation under a similar acceleration budget.

Compared with direct spatial caching, Cache3D provides a substantially better fidelity–efficiency trade-off. Token- or voxel-level reuse can achieve competitive acceleration, but introduces noticeably larger latent and geometric deviations. In contrast, Cache3D retains more of the full-inference output while avoiding direct spatial feature replacement.

The same trend is observed on Toys4K. Cache3D remains close to full inference across geometry and rendered-view metrics while providing substantial acceleration, demonstrating that its advantage is consistent beyond the example-image benchmark.

Qualitative Comparison. Fig. 3 compares Cache3D with full inference and existing step- and token-level caching methods. EasyCache and TeaCache generally retain the coarse object structure but introduce visible deviations in local details. The token-level adaptations produce more severe artifacts, including broken components, irregular boundaries, and noisy geometry. In contrast, Cache3D remains visually close to full inference, particularly around thin structures and small components that may be under-emphasized by globally aggregated mesh metrics. These results support our central design: localized cell-level drift detects spatially concentrated changes, allowing transformer computation to be restored before cache errors develop into visible geometric artifacts.

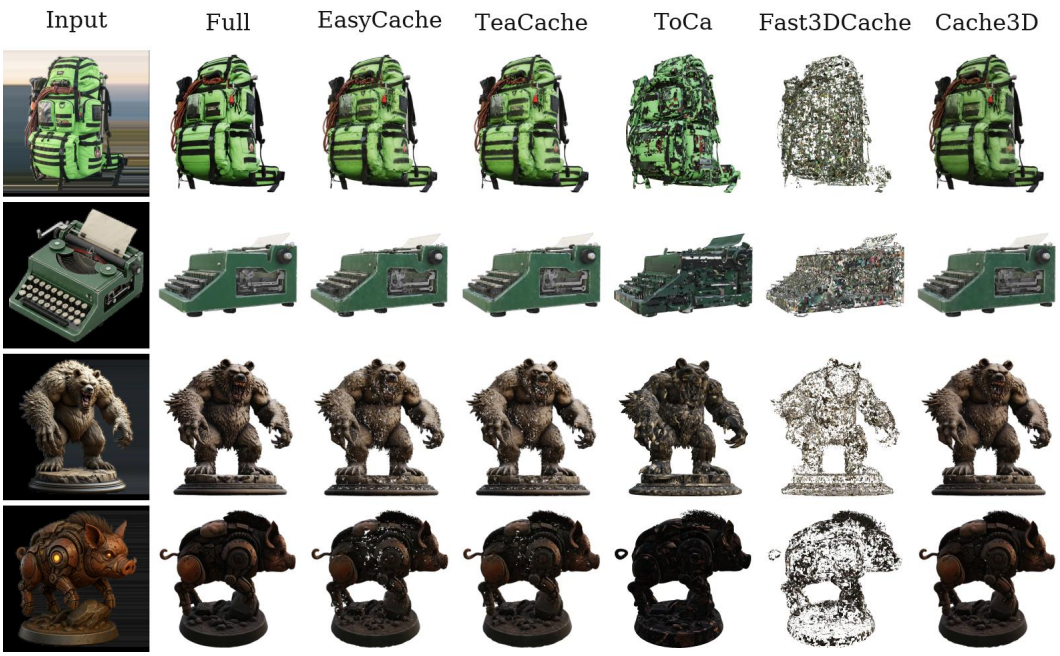

Figure 3: **Qualitative comparison with full inference and caching baselines.** Compared to prior methods, Cache3D better preserves the reference geometry and surface details of full inference without noticeable degradation. In contrast, token-level reuse baselines exhibit broken thin structures, corrupted surface topology, and severe artifacts.

Table 3: **Ablation study on the example-image benchmark.** We evaluate the contributions of spatial cell reliability, adaptive block selection, and block fallback. Quality metrics are measured against full inference.

Variant	Speedup ↑	Shape RMSE ↓	CD ↓	F@0.02 ↑	LPIPS ↓
Cache3D	1.633×	**0.3167**	0.00931	**0.9637**	**0.0561**
w/o Adaptive Selection	1.608×	0.3213	**0.00930**	0.9635	0.0575
w/o Block Fallback	1.600×	0.3861	0.00934	0.9632	0.0713
w/o Cell Reliability	**1.898**×	0.5676	0.00959	0.9552	0.1537
EasyCache	1.594×	0.3769	0.00934	0.9631	0.0681

Ablation Study. Tab. 3 and Fig. 4 evaluate the main components of Cache3D on the example-image benchmark. EasyCache is included as an external step-level baseline that uses neither our Cell Reliability nor Block Fallback, providing a reference for the overall efficiency–fidelity trade-off. The complete method achieves the lowest Shape RMSE and LPIPS and the highest F-score, while also providing a slightly higher speedup than EasyCache. These results demonstrate that the proposed reliability-guided execution policy improves fidelity without compromising acceleration.

Cell Reliability is essential for preventing unsafe reuse. Removing it increases the speedup from $1.633\times$ to $1.898\times$, but substantially increases Shape RMSE from 0.3167 to 0.5676 and LPIPS from 0.0561 to 0.1537, while reducing F@0.02 from 0.9637 to 0.9552. This shows that aggressive Step Reuse without localized reliability estimation skips more computation at the cost of considerably larger latent, geometric, and perceptual errors.

Block Fallback provides an effective intermediate mode between Step Reuse and Full Refresh. Removing it increases Shape RMSE from 0.3167 to 0.3861 and LPIPS from 0.0561 to 0.0713, while also reducing the speedup from $1.633\times$ to $1.600\times$. Thus, uncertain states do not always require a complete model evaluation. Selectively recomputing volatile transformer blocks can recover fidelity more efficiently than directly falling back to Full Refresh.

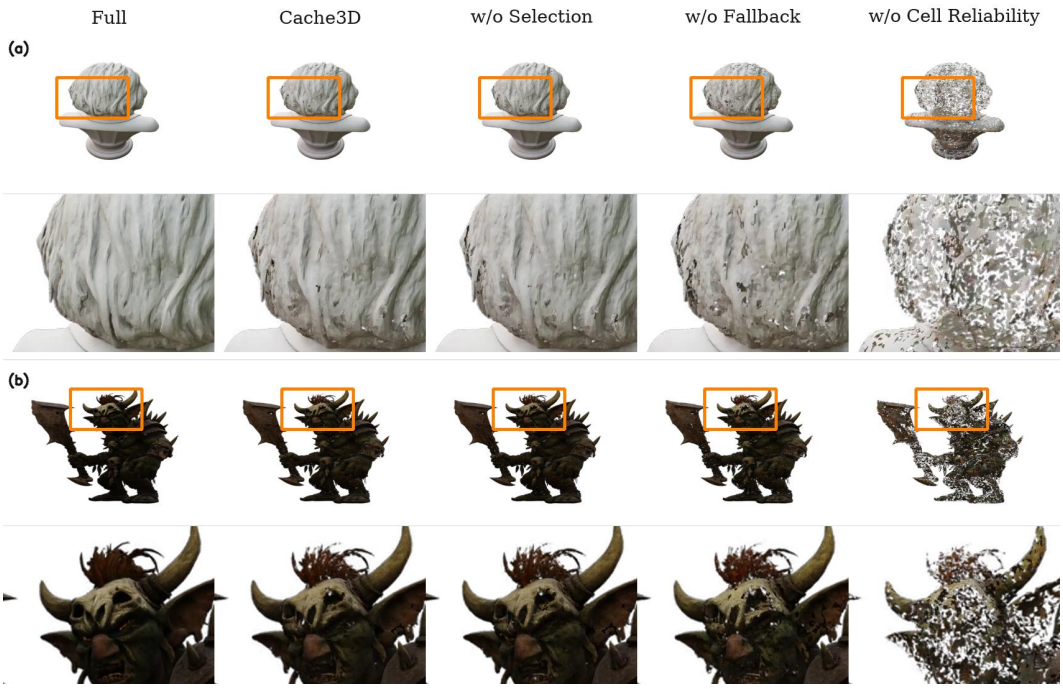

Figure 4: **Qualitative ablation of Cache3D.** Zoomed-in regions highlight the impact of removing individual design components against full inference (complementing Tab. 3). Removing the adaptive mode selection (*w/o Selection*) or the block fallback mechanism (*w/o Fallback*) causes localized blur and structural inaccuracies, while discarding the spatial cell pooling (*w/o Cell Reliability*) leads to severe fragmentation and surface noise.

We further replace Adaptive Selection with a fixed block policy. This variant remains competitive, confirming the benefit of block-level fallback, but produces slightly higher Shape RMSE (0.3213 vs. 0.3167) and LPIPS (0.0575 vs. 0.0561), together with a lower speedup ($1.608\times$ vs. $1.633\times$). Adaptive Selection therefore further improves the efficiency–fidelity balance by adjusting block reuse according to temporal residual variation.

Overall, the ablation demonstrates that the two control levels in Cache3D are complementary. Spatial Cell Reliability determines when whole-step reuse becomes unsafe, while Adaptive Block Fallback determines how much transformer computation should be restored. Together, they prevent excessive cache error without reverting every uncertain sampling state to a complete model evaluation.

5 Conclusion

In this work, we identify two obstacles to reliable caching in structured latent 3D generation. Global step-level signals can hide localized geometric changes, while token-level reuse can disrupt self-attention by mixing stale and recomputed representations. To address both issues, we introduce Cache3D, a training-free framework that separates spatial reliability estimation from computation reuse. Cache3D aggregates anchor-relative token drift within localized grid cells and uses the resulting risk to select among Step Reuse, Block Fallback, and Full Refresh. This hierarchical policy bypasses redundant evaluations when cached states remain reliable and progressively restores computation when localized changes emerge. Experiments on structured 3D generation benchmarks show that Cache3D achieves a favorable efficiency–fidelity tradeoff, better preserving geometric structure than existing step-level and token-level caching strategies. On the TRELLIS 2.0 example-image benchmark, Cache3D achieves a 1.633× end-to-end speedup and reduces shape-latent relative RMSE by 16.0% compared with EasyCache, while maintaining comparable geometric fidelity. These results support our central design principle: spatially localized reliability estimation can guide coarse-grained computation reuse without resorting to fragile token-wise output stitching.

AI USE STATEMENT

In this work, we used generative AI tools to assist with research brainstorming, language editing, document organization, and code debugging. All AI-assisted outputs were manually reviewed, revised, and, where applicable, validated through experiments by the authors. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

REFERENCES

Fanpu Cao, Yaofo Chen, Zeng You, and Wei Luo. Procache: Constraint-aware feature caching with selective computation for diffusion transformer acceleration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 19862–19870, 2026.

Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Yibo Chen et al. Delta-dit: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024a.

Zhaoxi Chen, Jiaxiang Tang, Yuhao Dong, Ziang Cao, Fangzhou Hong, Yushi Lan, Tengfei Wang, Haozhe Xie, Tong Wu, Shunsuke Saito, Liang Pan, Dahua Lin, and Ziwei Liu. 3dtopia-xl: High-quality 3d pbr asset generation via primitive diffusion. *arXiv preprint arXiv:2409.12957*, 2024b.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. In *Advances in Neural Information Processing Systems*, 2023a.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023b.

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Yushi Huang, Zining Wang, Ruihao Gong, Jing Liu, Xinjie Zhang, Jinyang Guo, Xianglong Liu, and Jun Zhang. Harmonica: Harmonizing training and inference for better feature caching in diffusion transformer acceleration. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 25835–25858. PMLR, 2025.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model, 2024. URL https://arxiv.org/abs/2411.19108.

Minghua Liu, Ruoxi Shi, Lizhen Chen, Zhuoyang Zhang, Chao Xu, Xiaoshuai Wei, Hansheng Chen, Wei Zeng, and Hao Su. Shapegpt: 3d shape generation with a unified multi-modal language model. *arXiv preprint arXiv:2311.17618*, 2023.

Jinming Lou, Wenyang Luo, Yufan Liu, Bing Li, Xinmiao Ding, Weiming Hu, Jiajiong Cao, Yuming Li, and Chenguang Ma. Token caching for diffusion transformer acceleration. *arXiv preprint arXiv:2409.18523*, 2024.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, 2022.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Zhengyao Lyu, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. Fastercache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations*, 2025.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. In *Advances in Neural Information Processing Systems*, 2024.

Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

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

Junxiang Qiu, Lin Liu, Shuo Wang, Jinda Lu, Kezhou Chen, and Yanbin Hao. Accelerating diffusion transformer via gradient-optimized cache. In *International Conference on Computer Vision*, pp. 17608–17617, 2025.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Stefan Stojanov, Anh Thai, and James M. Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Maxim Tatarchenko, Stephan R. Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

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*, 2023.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.

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. *Tech report*, 2025.

Mengyu Yang, Yanming Yang, Chenyi Xu, Chenxi Song, Yufan Zuo, Tong Zhao, Ruibo Li, and Chi Zhang. Fast3dcache: Training-free 3d geometry synthesis acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27030–27040, 2026.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

Xin Zhou, Dingkang Liang, Kaijin Chen, Tianrui Feng, Xiwu Chen, Hongkai Lin, Yikang Ding, Feiyang Tan, Hengshuang Zhao, and Xiang Bai. Less is enough: Training-free video diffusion acceleration via runtime-adaptive caching. *arXiv preprint arXiv:2507.02860*, 2025.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *International Conference on Learning Representations*, 2025.

A Cache States and Update Protocol

The method is training-free and leaves all pretrained TRELLIS 2.0 parameters unchanged. Within each structured-latent sampling stage, sparse token coordinates and sequence indices remain fixed. All cache states are cleared when transitioning between generation stages or whenever the sparse coordinate layout changes.

Step-level state. The step cache consists of the anchor latent $\mathbf{A}_\tau$ and the velocity–latent residual $\mathbf{r}_\tau = \overline{\mathbf{v}}_\tau - \mathbf{A}_\tau$. Step Reuse reads these buffers without modification. Both Full Refresh and Block Fallback produce a new velocity prediction and therefore synchronize the step-level state at the current sampling step.

Block-level state. For transformer block b, we cache the complete block transformation residual $\mathbf{R}_b = \mathbf{H}_b^{\text{out}} - \mathbf{H}_b^{\text{in}}$. This residual includes the combined transformation of the attention and feed-forward sublayers represented by the block output. During Block Fallback, stable blocks apply the cached residual to their current input, whereas volatile blocks are executed normally. Block residual caches and sensitivity scores are synchronized only during Full Refresh. This prevents partially refreshed block states from becoming mutually inconsistent reference points.

Cold start and stage transitions. At the start of a stage, the first Full Refresh initializes the anchor, step residual, all block residuals, and block sensitivity statistics. No cache is shared across Shape-SLat and Texture-SLat sampling. The cache is also purged if the sparse coordinate layout changes, although this does not occur within the fixed-coordinate rollout considered in our experiments.

B Implementation Details

We integrate Cache3D into TRELLIS 2.0 without modifying model weights or performing additional training. The pretrained transformer predicts velocity, and the latent is advanced with the Euler update given in Eq. (1). Shape-SLat and Texture-SLat use separate cache instances. Cell coordinates are computed once from the fixed sparse token coordinates of each stage and reused throughout that stage.

For both stages, we use cell width $s = 4$, mean pooling within occupied cells, and maximum pooling across cells. We set $(\gamma_{\text{fb}}, \gamma_{\text{full}}) = (0.30, 0.48)$, $G_{\min} = 3$, two warm-up steps, one terminal Full Refresh, and EMA momentum $\beta = 0.8$. Block selection considers all 30 transformer blocks, with tolerances 1.10 and 1.25 for Shape-SLat and Texture-SLat, respectively. Selected cached blocks use hard execution skipping in both stages. All experiments are conducted on a single NVIDIA A100 80GB GPU.

C Spatial Reliability Analysis

The cell statistic is a controller-side latent-drift signal rather than a calibrated predictor of decoded geometry or velocity error. We therefore test whether it can rank sampling states by the risk of Step Reuse. At each state in the diagnostic rollouts, we additionally evaluate the full model and measure the relative RMSE between the counterfactual Step-Reuse velocity and the full velocity. These teacher evaluations are used only for analysis and do not alter the executed trajectory or the reported runtime. Figure 5 compares this error with D_{cell}, while Figure 6 visualizes the spatial token drift and the occupied cell attaining the maximum response.

Both panels show an upward trend: states with larger cell risk generally incur larger error if the step-level cache is reused. The tighter Shape-SLat trend and the high Texture-SLat rank correlation support using D_{cell} to rank reuse risk and trigger fallback. The additional scatter in Texture-SLat shows that this signal is not a calibrated error predictor. The figure measures velocity error rather than decoded geometric error, and mode colors need not be perfectly separated because forced evaluations and the minimum evaluation gap also affect the execution decision.

Figure 6 illustrates how the scalar controller signal is constructed. Token-wise drift is first pooled over spatially local support so that a multi-token cell response represents aggregate local change. The

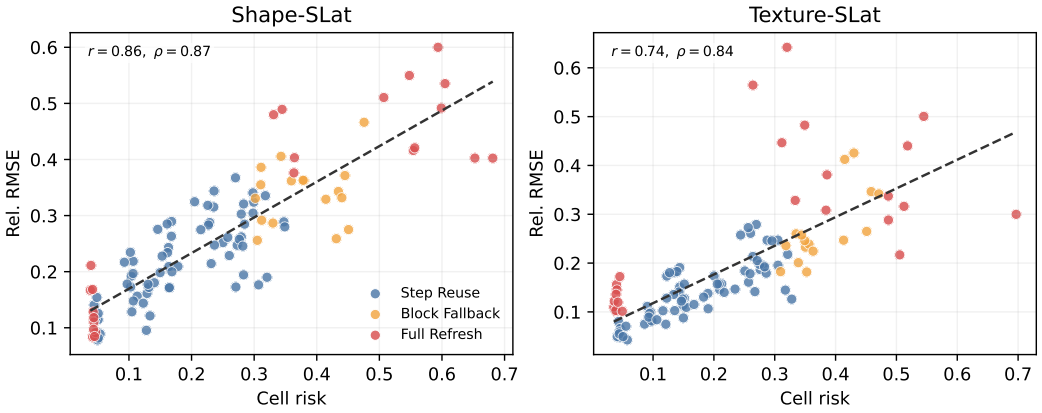

Figure 5: Cell risk versus Step-Reuse error. Each point is one sampling state, colored by the executed mode. The vertical axis is the relative RMSE between the velocity produced by counterfactual Step Reuse and a full model evaluation. Cell risk is strongly correlated with reuse error for Shape-SLat and remains rank-consistent but noisier for Texture-SLat . Low-risk Full Refresh points correspond to forced warm-up or terminal evaluations.

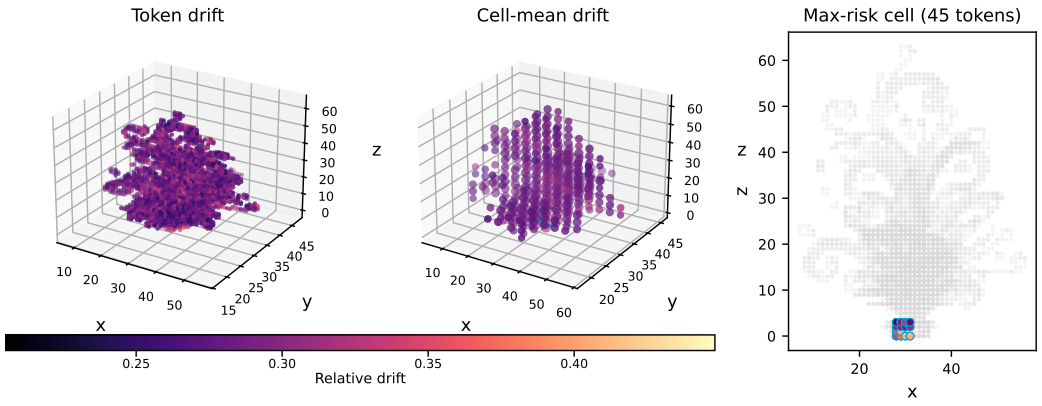

Figure 6: Spatial construction of the cell reliability signal at a representative Texture-SLat sampling state. Left: drift is measured for each sparse token. Middle: token drift is averaged within each occupied cell, and the maximum response is highlighted. Right: the 45 tokens supporting that response form a local spatial neighborhood. Cells are used only to estimate risk; execution remains unified.

largest cell-mean response is then used to govern the unified execution mode. The visualized cells are measurement neighborhoods rather than independently cached or recomputed token groups.

Because TRELLIS 2.0 uses a sparse coordinate layout, occupied cells may contain different numbers of tokens. Table 4 summarizes the occupancy distribution under the paper configuration. These statistics clarify the support over which the local mean in Eq. (4) is computed. An occupied cell contains 24.46 tokens on average, while 3.34% of occupied cells are singletons. Thus, intra-cell averaging acts on a multi-token neighborhood for most occupied cells; singleton cells naturally receive no within-cell smoothing. Texture-SLat is instantiated on the Shape-SLat sparse coordinates, so the two stages have identical occupancy statistics.

Comparison with Alternative Reliability Strategies. We compare Cache3D with a representative global step-level caching baseline and two controlled variants of its reliability aggregation. EasyCache is included as an external reference for global step-level reliability and differs from Cache3D in both its controller and reuse mechanism. Therefore, its gap should not be attributed solely to the

Table 4: Occupied-cell statistics for the frozen cell width $s = 4$.

Statistic	Shape-SLat	Texture-SLat
Tokens / asset	20478 ± 12651	20478 ± 12651
Occupied cells / asset	837 ± 378	837 ± 378
Mean occupancy	24.46	24.46
Median occupancy	23.0	23.0
P90 occupancy	48.0	48.0
Singleton cells (%)	3.34	3.34

Table 5: **Comparison with alternative reliability strategies.** EasyCache is an external global step-level baseline. The remaining variants retain the Cache3D controller and residual-reuse mechanism, changing only the reliability aggregation rule. Thresholds are adjusted to obtain comparable speedups. Lower Shape relative RMSE is better.

Reliability strategy	Speedup ↑	Shape Rel. RMSE ↓
Global step-level (EasyCache)	$1.594\times$	0.3769
Token Max	$1.617\times$	0.3268
Cell Mean + Top5	$1.626\times$	0.3252
Cell Mean + Max (Cache3D)	$1.633\times$	**0.3167**

reliability signal. For the controlled Token Max and Cell Mean + Top5 variants, we retain Cache3D's residual-reuse mechanism, three execution modes, and Block Fallback, changing only the aggregation rule. Their thresholds are adjusted to obtain comparable speedups.

As shown in Table 5, Token Max remains sensitive to localized changes, but its decision depends on a single token and can therefore respond to isolated fluctuations. Cell Mean + Top5 first suppresses token-level noise through local averaging, but aggregating several high-response cells weakens the influence of the most critical region. Cache3D instead takes the maximum over cell-averaged responses, preserving sensitivity to the strongest coherent local change. Under comparable acceleration, it reduces Shape relative RMSE by 3.1% and 2.6% relative to Token Max and Cell Mean + Top5, respectively. The controlled comparisons therefore support Cell Mean + Max as the most effective aggregation rule within the Cache3D execution framework.

Project Page. Additional synchronized turntable comparisons are available on our project page at `https://cache3d.github.io/Cache3D/`. The page compares full inference, Cache3D, EasyCache, and ToCa using the same input, seed, and fixed-distance camera trajectory, allowing localized geometric and appearance differences to be inspected across viewpoints.

D Baseline Adaptation Details

All baselines use the same input, random seed, sampler steps, numerical precision, and pretrained TRELLIS 2.0 weights as full inference. Cache states are reset between Shape-SLat and Texture-SLat stages. All baseline configurations are fixed before evaluation.

EasyCache and TeaCache. We apply their global, timestep-level cache decisions to the structured-latent transformer outputs. Their thresholds are fixed before evaluation and are shared across the benchmark.

ToCa (V-norm port). The original ToCa family caches features at transformer-layer or module granularity. Our TRELLIS 2.0 port maintains cache state separately for every block and module. Full computation is used for the initial three steps and then periodically every four steps. On cached steps, attention features are reused and selected MLP tokens are refreshed. Token importance uses the norm of the value representation because the FlashAttention path does not expose a materialized attention matrix. The base fresh ratio is 0.10, modulated by the layer-dependent factor $1.5 - b/(B - 1)$, and the soft cache-age weight is 0.25.

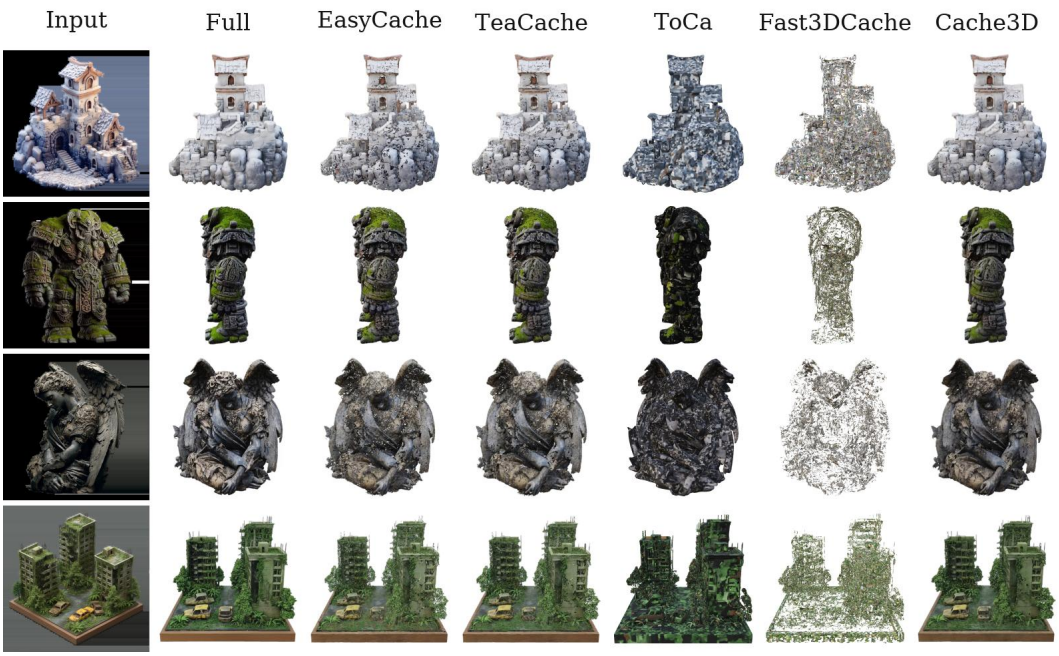

Figure 7: **Additional qualitative comparisons on the example-image benchmark (Part I).** The examples cover an architectural scene, two detailed sculptures, and a multi-building scene. Cache3D more closely follows full inference while preserving fine structures and overall appearance.

Fast3DCache-style token reuse. We transfer the SSC-based ranking, subset transformer execution, and stale–fresh output stitching mechanism to both Shape-SLat and Texture-SLat. The score combines min–max normalized velocity magnitude and acceleration with weights 0.3 and 0.7, respectively. The requested cache ratio is 70%. A token is forced fresh on its third consecutive cache request, and a full correction is performed every third step in the final phase. We refer to this baseline as *Fast3DCache-style token reuse* because the occupancy-based PCSC scheduler released for the dense geometry representation is not directly defined on the fixed sparse SLat coordinates used here.

Figures 7 and 8 provide additional comparisons on the example-image benchmark. All methods use the same input, random seed, and rendering protocol. These examples focus on comparisons with the step-level baselines, for which Cache3D remains visually closer to full inference in fine surface details, local geometry, and appearance. For completeness, each row also includes the ToCa and Fast3DCache token-level baselines rendered with the same camera pose, lighting, background, and resolution as the other methods.

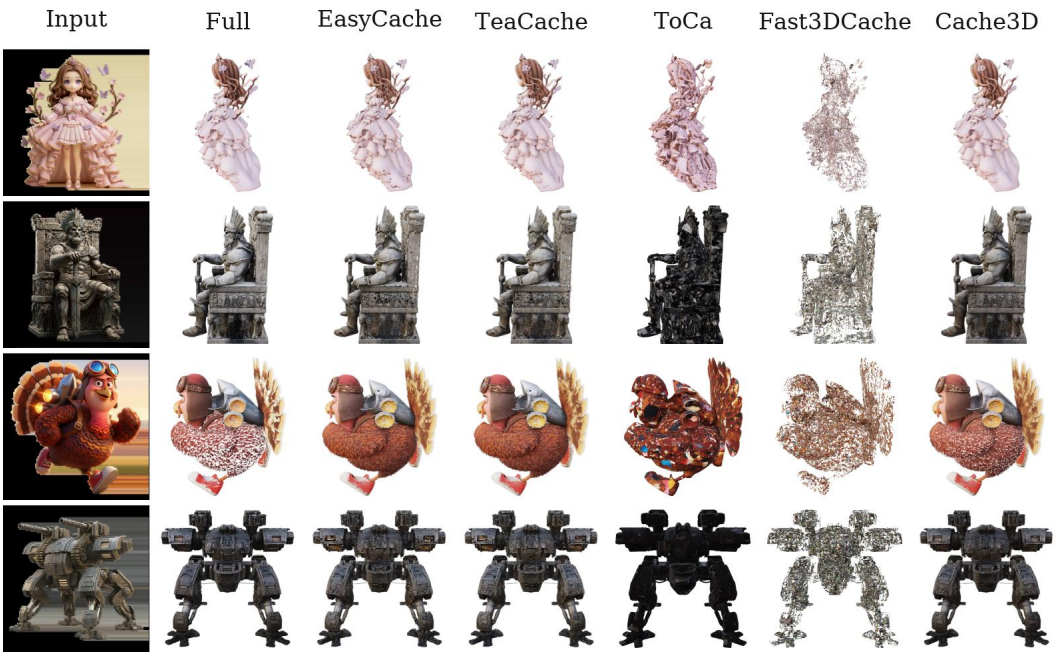

Figure 8: **Additional qualitative comparisons on the example-image benchmark (Part II).** These examples include a character with thin decorative elements, two detailed figures, and a mechanical asset. Cache3D remains close to full inference across these different structures.